Traffic congestion propagation identification method in smart cities

Attila M. Nagy, and Vilmos Simon

***Abstract*—Managing the frequent traffic congestion (traffic jams) of the road networks of large cities is a major challenge for municipal traffic management organizations. In order to manage these situations, it is crucial to understand the processes that lead to congestion and propagation, because the occurrence of a traffic jam does not merely paralyze one street or road, but could spill over onto the whole vicinity (even an entire neighborhood). Solutions can be found in professional literature, but they either oversimplify the problem, or fail to provide a scalable solution.**

In this article, we describe a new method that not only provides an accurate road network model, but is also a scalable solution for identifying the direction of traffic congestion propagation.

Our method was subjected to a detailed performance analysis, which was based on real road network data. According to testing, our method outperforms the ones that have been used to date.

***Index Terms*—congestion propagation, frequent propagation trees, traffic study, city planning**

I. INTRODUCTION

One of the major problems of traffic in big cities around the world is the phenomenon of traffic congestion (traffic jams) on the road network. Traffic jams have a serious effect not only on the lives of drivers, but also on every city inhabitant. Traffic jams increase not only energy and fuel consumption [1], but also harmful emissions [2]. According to a laboratory testing [3], congestion-related emissions cause increased cases of allergies and exacerbate existing conditions among people who are sensitive to it. Additionally, other research [4] shows that traffic jams raise the risk of heart attack. That is why attention needs to be paid to avoiding traffic jams and possibly eliminating them, because in addition to significant economic costs [5], they are also harmful to the health of city inhabitants (which can also be expressed as a financial costs for health insurers).

Intelligent city management systems can provide a solution to these problems, or at least substantially reduce the negative effects on the daily life of city dwellers. It is the task of these systems to continually monitor the traffic and to provide information on the basis of the collected data, as well as to manage the automated allocation of resources [6], for example, opening new lanes or closing them, adapting traffic lights to current traffic conditions [7] or assisting route planning applications with accurate forecasts.

Attila M. Nagy is with the Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Techonolgy and Economics, Hungary, e-mail: anagy@hit.bme.hu.

Vilmos Simon is with the Department of Networked Systems and Services, Faculty of Electrical Engineering and Informatics, Budapest University of Techonolgy and Economics, Hungary, e-mail: svilmos@hit.bme.hu.

DOI: 10.36244/ICJ.2021.1.6

In the first generation of the intelligent traffic management systems, the utilized data sources were different type of presence sensors in fixed positions, which were able to detect the presence of nearby vehicles. Initially, inductive loop detectors were the most popular, but nowadays, a wide variety of sensors became available such as traffic cameras, laser radar sensors or microwave radar sensors [8], [6]. Recently, the advent of GPS equipped smartphones and vehicles has given rise to a relatively new type of data source that could supplement presence type sensors to gather more detailed information or to get data about the roads, which have not been covered with presence sensors yet. In addition to existing data sources, the emerging Vehicle to Everything (V2X) communication technologies [9], [10] will become a vital data source in the future that can improve the performance of intelligent traffic management systems. The V2X defines messaging protocols to communicate between cars and the infrastructure. The V2X protocols are already integrated into the newest vehicles, but the number of equipped vehicles is still low; thus, the role of V2X will be more significant in future applications.

When a traffic jam starts to develop on a segment of the road network, it often also affects the surrounding road segments of the network. These effects are complicated - they always depend on current environmental factors (time of day, weather, holidays, etc.) but if we examine properly chosen time periods, we can collect useful information. The information gathered could make the operation of intelligent city management systems more efficient and provide new data to city planners and managers.

There are several studies exploring the propagation of the effects of developing traffic jams, which face a number of significant challenges. The first is that the road networks of large cities form an extensive and complex system in which the study of the traffic jam propagation is difficult to scale. Several studies have developed alternative models for solving the problem, but these oversimplify the road network, and so the output of their methods are imprecise. The second challenge is that it is not enough to detect a traffic jam, it is also important be able to identify the relationships between the road segments.

In this article, we would like to present a new method we have developed, which takes into account the spatial and temporal relationships:

1) is able to identify the appearance of congestion propagation across the entire road network in a scalable way,
2) is able to identify frequently occurring traffic jam propagation within these (the importance of which will be explained later).

The output of the process can be used by city traffic management system operators to map the source and propagation of frequently occurring traffic jams, and identify bottlenecks of the road network, which can be used in city traffic network planning and real-time interventions (if they have an Intelligent Transportation System (ITS) system). The output can be useful for route planning methods and can be used to refine forecasts of traffic prediction methods as well.

The remainder of this article contains the following chapters. Section II summarizes what research has been done in this field to date. In Section III, we introduce a new definition for traffic jams that, unlike the definition used in previous research, does not require manual parameter setting. We then introduce our developed congestion propagation detection method in Section IV and we perform a detailed performance analysis in Section V. We end the article with a short conclusion in the Section VI.

II. Related research

A. Congestion occurrence

To be able to reliably identify a traffic jam occurrence from traffic data, you first need to define exactly what a traffic jam is. Traffic jam definitions are not uniform in the literature [11], and they often depend to a large extent on empirically set parameters.

Methods found in the literature, which are used to explore the propagation of traffic jams or predict traffic congestions rely almost exclusively on the speed data to determine the size of a traffic jam. In addition, we have also found other definitions from related researches that use other types of data. The definitions can be separated into three main categories:

- speed-based methods,
- travel time-based methods,
- volume-based methods.

The speed-based methods can be separated to threshold-based and ratio-based subcategories. In the case of threshold-based methods [12], [13], [14], the researchers manually, in advance, determine the possible traffic states (typically 4-5 traffic classes) and their associated boundaries. Then the measured speed data is categorized into the defined traffic classes. As an extension, the authors of [15] defined the boundaries between traffic classes as fuzzy and also incorporated other coefficients such as traffic volume.

The ratio-based methods [16], [17], [18], [19], [20], [13], [21] do not determine velocity limits but numbers in a ratio. They compare the current measured speeds with the previously measured average or *free-flow* speeds. The *free-flow* is the speed at which the vehicle drivers would be able to travel if unimpeded by other vehicles [22]. In the literature, we have found two ways to determine the *free-flow* speed. The Speed Reduction Index (SRI) definition [23], [24] uses the 85th percentile of the off-peak speed, while the Speed Performance Index (SPI) definition [24] applies the maximum permissible road speed.

If the ratio of current speed to the average (*free-flow*) speed is lower than the pre-determined ratio, then the current measured speed is considered a traffic jam situation, that is, based on the velocities measured in the area, they determine that a traffic jam has occurred. Some methods [16], [13], [24] define more than one traffic class. In these cases, different ratios are established for every traffic class.

The travel time-based methods use probe vehicle data to determine the congestion level of the examined road segments. The Travel rate method [25] calculates the ratio of the current segment travel time and the segment length to quantify the congestion level. The Delay rate [26], [25] and Delay ratio [26], [25] methods are the extensions of the Travel rate method where the measured travel rate is compared with a predefined acceptable travel rate value. The Relative Congestion Index (RCI) definition [27] relies on the ratio of the current measured travel time and the *free-flow* travel time. The *free-flow* travel time can be calculated with the ratio of the road segment length and *free-flow*.

The volume-based methods examine the traffic volume that denotes the number of passing vehicles in predefined time frames. A well-known volume-based method is the Volume to Capacity (V/C) ratio definition [28] that compares the measured volume with the maximum number of vehicles that a segment can handle within its capacity. The V/C ratio is often used together with the Level of Service (LoS) [29] where the measured V/C ratio values are classified in 6 traffic classes.

Using the criteria of a good congestion measure from article [11], the majority of these methods' weak point is that they only take one data type into consideration, and except for some definitions [24], [28], do not describe the state of the traffic appropriately because the occurrence of the traffic jam always depends on the properties of the road segment as well. For example, the number of lanes, the speed limits for a given segment of road, the number of vehicles passing through, or the road segment's capacity all affect whether or not a traffic jam has occurred.

Another problem that appears is if the method uses manually set values or values that rely on previously measured data. Unfortunately, several cases occur where an explanation is missing as to why a threshold value or a ratio is used, seemingly depending on the subjective decision of the authors.

B. Congestion propagation

As opposed to the occurrence of a traffic jam, the definition of the propagation of a traffic jam is mostly uniform in the professional literature. A road segment can be in one of two states: congested (1) or free flowing vehicle movement (0). Traffic jam propagates between neighboring road segments A and B, if at time t road segment A is congested and B isn't, but at time $t+1$ segment B is congested too.

The *Propagation Probability Graph (*PPG*)* [20] method models the road network as a directed graph. It relies on a historical database to decide on the probability of traffic jam propagation between two neighboring road segments. It states that the propagation of the traffic jam has a Markov property. In this context, this means that the likelihood of a traffic jam propagating between road segments A and B is independent of what traffic propagation probabilities have been measured before road segment A. The PPG method uses this property

to assign probabilities to each traffic jam propagation path. It then only pays attention to whichever possibility is greater than a predetermined value γ. PPG does not take into account those situations where other road segments than A also flow into road segment B, and therefore in these cases the results can be imprecise.

The *Congestion Prediction Model with ConvLSTM (*CPM-CONVLSTM*)* [30] method also models the road network with a directed graph. It collects propagation patterns from the road network, but unfortunately does not specify how it does this. It places a square grid over the road network, and then maps the propagation. Thus, the propagations can be described by directed edges between the points of the square grid. The CPM-CONVLSTM method, like PPG, also focuses on predicting traffic jam propagations. Using the square grid model, they train a Convolutional Long Short Term Memory (CONVLSTM) network [31]. A major disadvantage of this method is that the square grid-based model oversimplifies the road network, and this simplification makes the modeling inaccurate. For example, if a busy highway and another nearby unfrequented road are in the same cell they cannot be distinguished despite the fact that they have completely different traffic demand.

The *Cascading Patterns in Scale-Free Network (*CP-SFN*)* [32] method also models the road networks as a directed graph. This method's purpose is to find propagation graphs in a graph of the road network. These propagation graphs will be subgraphs of the road network graph. It considered two components to determine propagation paths: Individual Transmission Likelihood (ITL) and Environmental Intensity Inference (EMT). The ITL determines the probability of traffic jam propagation between two road segments. The monotonic exponential model used in social networks [33] was used to model propagation. The EMT component collects environmental information for the study, which is than used to weigh the output of the ITL. The EMT takes into account the Point of Interest (POI) that are close to the road segments and what weather conditions have been measured in the studied areas. The search for propagation patterns is carried out by the authors of the article using their own approximation procedure, since the Network Inference problem used in the article is an NP-hard problem [34]. The disadvantage of this method is that the original monotonic exponential model applied by CP-SFN operates on scale-free [35] graphs, while the road networks form a scale-rich [35] graph.

The aim of the *Spatio-Temporal Outlier (*STO*)* [36] method is to find frequent traffic jam propagation trees in the road network. The authors of the article did not study the road network as a graph, but divided the road network into regions using the Connected Components Labeling (CCL) process [37]. Relationships between regions were determined based on vehicle trajectories. The STO method identifies congestion as an outlier, and therefore when examining propagations it follows the propagation of outliers. It constructs congestion propagation graphs from the propagations using a recursive approach. The disadvantage of the STO method is that the regional division oversimplifies the road network. To overcome this problem, the authors of [38] implemented an extension of STO in which they already use the directed graph of the road network instead of regions. The other disadvantage of the STO method is that due to the search for recursive propagation trees its complexity is $\boldsymbol{O}(N^{T-1})$, where T is the length of the studied period in time intervals and N is the number of congestion phenomena in the period of time T.

The *Spatio-Temporal Congestion (*STC*)* [39] process develops the STO method further. Instead of using regions it models the road network as a directed graph of road segments. Congestion propagations are described by a directed graph of the road segments involved in the propagation that form a directed tree. It uses a new approach to search for congestion propagations, which examines the congestion trees backwards in time, so that the complexity of the procedure is only $\boldsymbol{O}(TN^2)$, where T is the length of the examined period in time intervals and N is the number of traffic jam phenomena during period T. It then filters out frequent congestion propagation trees from the collected propagation trees using the Apriori algorithm [40]. A congestion propagation tree counts as frequent if its occurrence is above an ϵ threshold in the studied time interval, so its frequency is greater than a predefined threshold value. Within a frequent congestion propagation tree, a propagation path is a directed path that connects the root of the tree to any leaf.

Although STC is significantly faster than the STO method, the $\boldsymbol{O}(TN^2)$ complexity remains too high for modeling a city with an extensive road network. The other problem is that the Apriori algorithm has exponential complexity $\boldsymbol{O}(2^M)$, where M is the number of congestion propagation found.

The *Spatio-Temporal Congestion Subgraph (*STCS*)* [41] method is a further development of the STC in which the FP-Growth algorithm [42] is used instead of the Apriori algorithm. Because the complexity of the FP-Growth algorithm is lower than that of the Apriori algorithm, its use made the search for frequent congestion propagation trees quicker, but STCS uses STC found propagation trees, which has $\boldsymbol{O}(TN^2)$ complexity.

After reviewing the literature, it can be seen that the problem is actively studied by many researchers and they have presented several methods to solve the problem of finding frequent congestion propagation trees.

The problem with these solutions is that either the models of the road network are overly simplified (for example, with a square grid), or the method's complexity is so high that in a real environment it is not able to search effectively or find every propagation in a sprawling city. They can not be used this way in real time, but even if the speed of the algorithms is not critical, they still require a lot of processing, which can be costly.

In the following chapters we introduce our newly developed method that solves the search for traffic jam propagation trees, in a way that:

1) describes the road network as a directed graph in the greatest detail possible,
2) while the algorithmic complexity is only linear, and therefore surpasses the existing solutions.

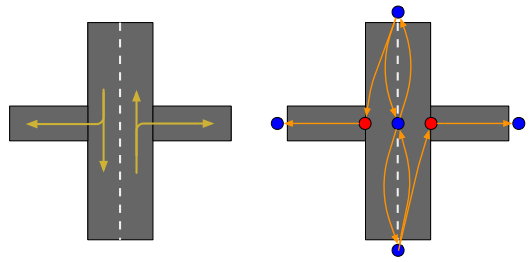


Fig. 1: Example for modeling a common intersection.

III. FLOW-SPEED RATIO-BASED CONGESTION DEFINITION

In Section II-A we have seen that there is currently no uniform traffic jam definition that does not use manually preset values. In this section, before introducing the Spatial Congestion Propagation Patterns (SCPP) algorithm, we thought it necessary to introduce a new traffic jam definition that does not require manual setting of thresholds or ratios, but does so automatically. Using our definition, it is easier and more reliable to generate input data for the Spatial Congestion Propagation Patterns (SCPP) algorithm.

In a real environment, the correct interpretation of a traffic jam always depends on the current road segment. Each road segment has a capacity value that determines how many vehicles can pass through that road segment per hour. The capacity is the theoretical upper limit of the number of vehicles that the measured hourly vehicle number never exceeds. If a higher load is imposed on a road segment it will mean that as the number of vehicles increases, their speed will begin to decrease. This will cause an increase in the ratio of traffic flow (volume) and speed until it finally reaches a critical value above which the traffic phenomenon can be considered a traffic jam. The critical flow-speed rate can be determined by the ratio of the capacity and the upper speed limit of the road segment.

Let $\boldsymbol{\mathcal{N}}(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{R}})$ be a directed graph representing a city's road network, where $\boldsymbol{\mathcal{I}} = \{I_1, I_2, \ldots, I_{|\boldsymbol{\mathcal{I}}|}\}$ is the set of intersections and $\boldsymbol{\mathcal{R}} = \{R_1, R_2, \ldots, R_{|\boldsymbol{\mathcal{R}}|}\}$ the set of road segments. Since the road network model uses directed edges, the bidirectional road segments will be represented with two directed edges. It is not uncommon to see that vehicles can turn only in specific directions in complex intersections. To model these intersections, additional intermediate nodes (intersections) have to be added. On Figure 1, we show an example for modeling a common intersection. The blue dots denote the intersections as the nodes of the graph, while the orange arrows are the directed edges. The red dots depict the intermediate nodes. These intermediate nodes help to model the allowed turning directions properly.

Using the recommendation of the Highway Capacity Manual 2016 (HCM2016) [43] and considering the current speed limit, the theoretical capacity of a segment of R_r (Free-way (FW)) road is described in Equation 1, where $S_{R_r,limit}$ is the maximum allowed speed in mph for road segment R_r, while the $Lanes$ is the number of lanes on the road segment R_r. This equation is only valid for freeways, whereas equations for multi-lane highways, signalized highways and other road types can be found in HCM2016 [43]. It is worth mentioning that the spread of autonomous vehicle will change the capacity formula in the future because the throughput of road segments will be increased due to better resource utilization [44], [45], [46].

Let $\boldsymbol{\mathcal{F}}_{\boldsymbol{R_r}} = \{F_1, F_2, \ldots, F_T\}$ be the time series of vehicle numbers (volume), where R_r is the edge of the road network graph $\boldsymbol{\mathcal{N}}(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{R}})$, where the data was measured, T is the length of the time series, and $F_t \in \mathbb{R}^+$ $(t = 1, 2, \ldots, T)$. The vehicle speed time series is given by $\boldsymbol{\mathcal{S}}_{\boldsymbol{R_r}} = \{S_1, S_2, \ldots, S_T\}$,where R_r is the edge of the road network graph $\boldsymbol{\mathcal{N}}(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{R}})$, where the data was measured, T is the length of the time series, and $S_t \in \mathbb{R}^+$ $(t = 1, 2, \ldots, T)$.

Using time series $\boldsymbol{\mathcal{F}}_{\boldsymbol{R_r}}$ and $\boldsymbol{\mathcal{S}}_{\boldsymbol{R_r}}$ the road segment's $\boldsymbol{\mathcal{FSR}}_{\boldsymbol{R_r}} = \{FSR_1, FSR_2, \ldots, FSR_T\}$ flow-speed rate time series can also be calculated. The instantaneous rate FSR_t of a road segment R_r can be determined using the following formula:

$$FSR_t = \frac{F_t}{S_t} \qquad t = 1, 2, \ldots, T \qquad FSR_t \in \mathbb{R}^+. \tag{2}$$

The critical flow-speed rate value for the road segment R_r:

$$FSR_{R_r,critical} = \frac{CP(S_{R_r,limit})}{S_{R_r,limit}}, \tag{3}$$

where $CP(S_{R_r,limit})$ is the capacity of the road segment and $S_{R_r,limit}$ is the speed limit of the road segment in mph. Critical flow-speed rate $FSR_{R_r,critical}$ is a predetermined constant value that does not vary with time.

Using the instantaneous and critical flow-speed rate, the current level of the traffic jam $\boldsymbol{\mathcal{C}}_{\boldsymbol{R_r}} = \{C_1, C_2, \ldots, C_T\}$ can be determined, where

$$C_t = \frac{FSR_t}{FSR_{R_r,critical}} \qquad t = 1, 2, \ldots, T \qquad D_t \in \mathbb{R}^+. \tag{4}$$

Definition 1. *At time t a traffic jam will occur on the road segment R_r of the road network $\boldsymbol{\mathcal{N}}(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{R}})$, if $C_t \geq 1$ ($C_t \in \boldsymbol{\mathcal{C}}_{\boldsymbol{R_r}}$) so $FSR_t \geq FSR_{R_r,critical}$.*

Figure 2 shows an example of a flow-speed rate-based congestion definition. Data shown in the figure were obtained from the Caltrans Performance Measurement System (PEMS) [47]. The first and second rows of the figure show vehicle number and speed data. The third row contains the momentary congestion levels. The dashed red line is the critical flow-speed rate of congestion. Times above the line can be considered a traffic jam. It is worth noting that in all periods where the speed has declined significantly, the congestion level is above the critical value.

$$CP(S_{R_r,limit}) = min(2200 + 10(S_{R_r,limit} - 50), 2400) \times Lanes, \tag{1}$$

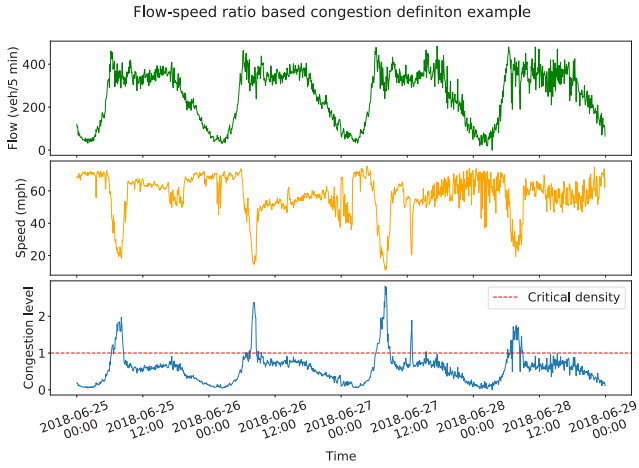


Fig. 2: Example for flow-speed ratio based congestion

This means that the flow-speed ratio-based method identifies traffic jams at the same time intervals as the speed-based methods. This is an important result, because while most of the methods from the literature require manually adjusting the parameters for each segment of the road, the parameters of the flow-speed ratio-based method are set automatically and an adaptive manner for each segment of the road.

In the case of propagation examining methods, it is often enough to know if the traffic jam had occurred, the actual level is not important (how much it is below or above the threshold). The momentary level of the congestion $\boldsymbol{\mathcal{C}_{R_r}}$ can be converted to a $\hat{\boldsymbol{\mathcal{C}}}_{\boldsymbol{R_r}} = \{\hat{C}_1, \hat{C}_2, \ldots, \hat{C}_T\}$ time series, where R_r is a road segment in the road network graph $\boldsymbol{\mathcal{N}}(\boldsymbol{\mathcal{I}}, \boldsymbol{\mathcal{R}})$, T is the length of the time series, and $\hat{C}_t \in \{0, 1\}$.

The $\mathcal{Z} : \boldsymbol{\mathcal{C}_{R_r}} \rightarrow \hat{\boldsymbol{\mathcal{C}}}_{\boldsymbol{R_r}}$ transformation can be used, where $\forall C_t$ can be converted to $\hat{C}_t$ as:

$$\hat{C}_t = \begin{cases} 1 & \text{, if } C_t \geq 1 \\ 0 & \text{, if } C_t < 1. \end{cases} \tag{5}$$

We wanted to validate the correctness of our definition, so we compared the binary output with other definitions from the literature. In our investigations, the SRI and SPI methods were implemented, and the used dataset was collected from the PeMS. The SRI required the precalculation of free-flow speed parameter, which were set to the 85th percentile of the off-peak speed. The SPI's max speed parameter was set to upper speed limit. In the examined dataset, the result of the Flow-speed ratio method differed from the SPI's output by only 2.8%, while 2.9% was the difference in the case of SRI. It is a minimal difference that means the Flow-speed ratio method gives appropriate output.

We have also checked the seven criteria of a good congestion measure from [11], and we have seen that the flow-speed ratio-based method satisfies all the listed criteria. The great advantage of the flow-speed ratio-based method is that it does not require the manual tuning of parameters for each segment of the road, and utilizes two data types for the calculation of the congestion level. Thanks to the used capacity function, our method adapts to the parameters of the given road segment. Using flow and speed data together could also be a disadvantage because the flow-speed ratio-based congestion definition cannot be used if only the flow, the speed, or the travel time data is available alone.

IV. Spatial Congestion Propagation Patterns (SCPP) algorithm

In this chapter, we will explain the operation of the Spatial Congestion Propagation Patterns (SCPP) algorithm in detail. The SCPP algorithm is able to solve the problem of searching for frequent congestion propagation paths with linear complexity while describing the road network in as much detail as possible using a directed graph. The presented algorithm is a brand new algorithm, and we did not use parts from other algorithms in the literature. We would like to highlight that the presented algorithm supports arbitrary congestion definition, which can produce binary congestion information (signalling if congestion occured or not). It means that the SCPP algorithm can also be used when the Flow-speed ratio-based congestion definition is not applicable.

First we define the necessary concepts, followed by the pseudo-codes and an explanation of the two main components of the algorithm in the order the processes are run:

1) Propagation Tree (PT) method: From the measured traffic data, it determines how many times the congestion propagated between road segments, and creates a describing graph based on the propagations, in which the nodes are the road segments and the edges are the propagations between the road segments.
2) Frequent Propagations (FP) method: Using the output of the Propagation Tree (PT) method and a certain threshold value, it searches for the most frequent traffic jam propagation patterns in the road network.

To run the SCPP algorithm, you need three input parameters: data matrix $\overline{\overline{\mathcal{C}}}_{\mathcal{N}}$, edge-adjacency matrix $\overline{\overline{\mathcal{A}}}_{\mathcal{N}}$, and a ϵ threshold value.

$\overline{\overline{\mathcal{C}}}_{\mathcal{N}}$ is a $|\mathcal{R}| \times T$ matrix containing the traffic jam observations of road segments $\mathcal{R}$ within road network $\mathcal{N}(\mathcal{I}, \mathcal{R})$ for a period of time T. Every field in the matrix has a value of 0 or 1 ($\overline{\overline{\mathcal{C}}}_{R_r,t} \in \{0, 1\}$, if $R_r \in \mathcal{R}$), where 1 means there is a traffic jam and 0 is a state of free-flow.

The matrix $\overline{\overline{\mathcal{A}}}_{\mathcal{N}}$ is the $|\mathcal{R}| \times |\mathcal{R}|$ sized edge-adjacency matrix of directed graph $\mathcal{N}(\mathcal{I}, \mathcal{R})$, which can be determined from the adjacency matrix of the line graph of the graph $\mathcal{N}(\mathcal{I}, \mathcal{R})$. $\overline{\overline{\mathcal{A}}}_{R_u,R_v} = 1$ if two road segments are adjacent to each other, otherwise $\overline{\overline{\mathcal{A}}}_{R_u,R_v} = 0$ ($R_u, R_v \in \mathcal{R}$).

Here ϵ is the threshold value that determines the minimum frequency. It is worthwhile to determine its value based on the length of the examined time period T, but it also depends on what qualifies as a frequent traffic jam in the environment, so designers can set it themselves depending on the traffic control and optimization goals.

Before beginning the presentation of methods PT and FP, we need to define what exactly "congestion occurrence" and "congestion propagation" are.

Definition 2. *Let R_v ($R_v \in \mathcal{R}$) be any segment of the road network $\mathcal{N}(\mathcal{I}, \mathcal{R})$. Let $\overline{\overline{\mathcal{A}}}_{\mathcal{N}}$ be the $|\mathcal{R}| \times |\mathcal{R}|$ edge-adjacency matrix of the road network $\mathcal{N}(\mathcal{I}, \mathcal{R})$. **Congestion occurence** is observed on the segment R_v at time $t+1$, if R_v was not congested at time t ($\overline{\overline{\mathcal{C}}}_{R_v,t} = 0$) but is by the time $t+1$ ($\overline{\overline{\mathcal{C}}}_{R_v,t+1} = 1$). In addition, none of the neighbors of road segment R_v were congested at time t, so for $\forall R_u$ it is true that $\overline{\overline{\mathcal{C}}}_{R_u,t} = 0$ if $R_u \in \mathcal{R}$ and $\overline{\overline{\mathcal{A}}}_{R_u,R_v} = 1$.*

Definition 3. *Let R_u and R_v ($R_u \neq R_v$; $R_u, R_v \in \mathcal{R}$) be the two road segments of road network $\mathcal{N}(\mathcal{I}, \mathcal{R})$. Let $\overline{\overline{\mathcal{A}}}_{\mathcal{N}}$ be the $|\mathcal{R}| \times |\mathcal{R}|$ edge-adjacency matrix of road network $\mathcal{N}(\mathcal{I}, \mathcal{R})$. **Congestion propagation** is observed between R_u and R_v at time $t+1$, if R_u was congested at time t ($\overline{\overline{\mathcal{C}}}_{R_u,t} = 1$) while R_v was not ($\overline{\overline{\mathcal{C}}}_{R_v,t} = 0$), but by time $t+1$ it is ($\overline{\overline{\mathcal{C}}}_{R_v,t+1} = 1$). R_u and R_v are adjacent to each other ($\overline{\overline{\mathcal{A}}}_{R_u,R_v} = 1$).*

*A. Propagation Tree (*PT*) method*

The Propagation Tree (PT) method uses data matrix $\overline{\overline{\mathcal{C}}}_{\mathcal{N}}$ and the edge-adjacency matrix $\overline{\overline{\mathcal{A}}}_{\mathcal{N}}$ as input to generate a directed tree describing the propagation path $\mathcal{P}(\mathcal{V}, \mathcal{L})$, which contains all the congestion propagation paths according to Definition 3 observed within the data matrix. Propagation paths are directed paths within a propagation tree that connect the root of the tree to any leaf.

$\mathcal{V} = \{V_0, V_1, V_2, \ldots, V_{|\mathcal{V}|}\}$ is the set of vertices of the tree in which any vertex $V_v = \{R_x \rightarrow \ldots \rightarrow R_y\}$ contains an observed unique propagation path. For example, let $V_1 = \{R_1 \rightarrow R_2\}$ be an observed propagation path. There is a directed edge from V_1 to V_v, if $V_v = \{R_1 \rightarrow R_2 \rightarrow R_r\}$, so that road segment R_r is adjacent to the road segment R_2 according to the edge-adjacency matrix $\overline{\overline{\mathcal{A}}}_{\mathcal{N}}$ and propagation path V_v is observable in data matrix $\overline{\overline{\mathcal{C}}}_{\mathcal{N}}$.

The root of the directed tree describing propagation paths $\mathcal{P}(\mathcal{V}, \mathcal{L})$ will be vertex $V_0 = \{-1\}$, which does not contain a valid propagation path. Edge L_l exists between V_0 and any vertex $V_v = \{R_r\}$ if a traffic jam occurrence according to Definition 2 or a traffic jam propagation according to Definition 3 has been observed on road segment $R_r \in \mathcal{R}$. In addition, each edge L_l records a *freq* value that describes the frequency of propagation. Everytime when a propagation is detected between V_u and V_v, *freq* value of L_l is increased by one (L_l is the directed edge from V_u to V_v).

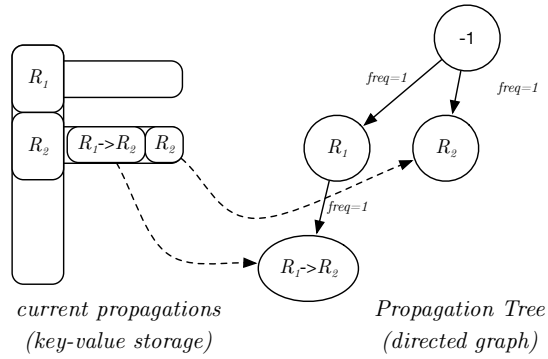


Fig. 3: Example of the relationship between $current\_propagations$ and $tree$ variables.

The steps of the Propagation Tree (PT) method are as follows. In the first step, we query the dimensions of the input data matrix, from which we determine the number of road segments (variable R) and the lenght of the examined period (variable T) (*line 1*). We then initialize the variables that store the results (*lines 3-5*). The $tree$ stores the propagation paths observed so far in a directed graph. In addition to the $tree$, an important variable is $current\_propagations$, which records which propagation paths are currently active. The variable is a key-value storage in which the keys are the identifiers of each road segment and the values are the propagation paths in which the road segment in the key is the last road segment of the propagation path. This way, if the traffic jam propagates from the road segment in the key it is possible to follow which propagation paths need to be further developed. Figure 3 is an example of the relationship between the $current\_propagations$ and $tree$ variables when an $R_1 \rightarrow R_2$ propagation happens.

PT starts processing the matrix $data\_mx$ forward in time (*lines 7-38*). The c_state variable stores the state at the current time (*line 9*), while the p_state variable contains the previous state. The $where$ method gives us the identifiers of the road segments on which the traffic jam phenomenon was observed by returning the array indices where the value is 1.

In the first loop, the previous state (p_state) will still be empty, so we simply copy the value of the current state (*lines 10-14*) to the tree and $current\_propagations$ variables using the $add\_propagations$ method (Algorithm 2) and then go to the second loop immediately.

Starting from the second loop, we iterate over the road segments that are congested in their current state (*lines 15-*

27). If the congested road segment (c_road) was not present in the previous state, it is worth investigating, because this is when important state change occurs (*lines 17-26*). We then extract the adjacent road segments ($source\_roads$) preceding the examined road segment from the edge-adjacency matrix adj_mx and see if we find them in the previous state (*lines 18-19*).

If none are found, it means that we have detected a congestion occurrence on the examined road segment (*lines 20-22*). Propagations that occur within the loop are stored in the $propagations$ key-value variable (*line 15*), where the key is the source of the propagation and the value is the road segment to which the traffic jam has propagated. In the case of a new propagation, the source of the propagation is the -1 identifier (*line 21*).

Otherwise, the congestion has not occurred on the examined road segment, but on one of the neighboring road segment (*lines 22-25*). Because it cannot be ruled out that several congested neighbors are affecting it simultaneously, all possible cases must be added to the $propagations$ variable.

Once we have traversed all the road segments from c_state, we add the propagations registered in $propagations$ to the $tree$ and $current\_propagations$ variables using the $add\_propagationsmethod$ (*lines 28-30*).

We then go through the road segments that were congested in the previous state and see which of them are still congested in the current state (*lines 32-36*). If the traffic jam on a road segment has ended, the propagation pointers of the road segment are removed from $current\_propagations$.

Since we have to study all road segments where a congestion phenomenon has occurred within all time intervals, the step number of the PT method is $\boldsymbol{O}(T|\boldsymbol{\mathcal{R}}|)$, where T is the length of the examined period in time intervals and $|\boldsymbol{\mathcal{R}}|$ the number of road segments, which is the upper estimate for the number of traffic jams that have occurred.

Another important part of the PT method is the $add\_propagations$ submethod. Its function is to maintain the tree describing the propagation ($tree$) and the currently tracked traffic jams ($current\_propagations$). When these variables need to be updated, $add\_propagations$ receives these two variables as input parameters and modifies their internal state.

As the first step, $add\_propagations$ checks to see if the source of the propagation ($from\_road$) obtained as a parameter is the same as node -1. If so, the propagation is added to the tree as a congestion occurrence (*lines 1-10*). In this case, it adds the target road segments to the tree variable if they do not exist yet (*lines 3-6*) and then increments the associated frequency $freq$ counter by one (*line 7*). It then registers the node in the $current\_propagations$ variable as a one-element propagation path identifier (*line 8*).

If the traffic jam has not occurred on that segment of road, it should be added to the tree as a propagation (*lines 10-30*). We have to iterate over all the cases where the source of the propagation was previously active. These are contained in $current\_propagations[from\_road]$ (*line 11*). Since the traffic jam can propagate from one road segment to several road segments at the same time, we have to go through these as well (*lines 12*).

Input: $data\_mx$:data matrix $\overline{\overline{\boldsymbol{\mathcal{C}}}}_{\boldsymbol{\mathcal{N}}}$;
adj_mx:edge-adjacency matrix $\overline{\overline{\boldsymbol{\mathcal{A}}}}_{\boldsymbol{\mathcal{N}}}$
Output: Directed graph describing the propagations $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{L}})$

```
1  R, T = data_mx.shape
2
3  current_propagations = empty
4  tree = DirectedGraph()
5  tree.add_node(−1, route_id = −1)
6
7  p_state = None
8  for t = 1 to T do
9      c_state = where(data_mx[:, t] == 1)
10     if p_state is None then
11         add_propagations(−1, c_state, state
           current_propagations, tree)
12         continue
13     end
14     propagations = empty
15     for c_road in c_state do
16         if c_road not in p_state then
17             pred_roads = where(adj_mx[:
                , c_road] == 1)
18             source_roads =
                intersect(pred_roads, p_state)
19             if len(source_roads) < 1 then
20                 propagations[−1].append(c_road)
21             else
22                 for s_road in source_roads do
23                     propagations[s_road].append(c_road)
24                 end
25             end
26         end
27         for from_road, to_roads in propagations do
28             add_propagations(from_road, to_roads,
               current_propagations, tree)
29         end
30
31         for c_road in p_state do
32             if c_road not in c_state then
33                 current_propagations[c_road] = []
34             end
35         end
36         p_state = c_state
37 end
```

Algorithm 1: The pseudo code of the PT method

We generate a pointer for the propagation path, which has occurred from the concatenation of the previous propagation path identifier and the congested road segment (*line 13*). If the propagation path identifier has not yet been included, we add it to the tree as a new node (*lines 14-16*), then increase the frequency counter (*line 17*) and register the propagation path identifier in the $current\_propagations$ variable to the to_road road segment (*line 18*).

Input: $from\_road$: source of the propagation; to_roads: target of propagation, $current\_propagations$: currently congested road segments, $tree$: tree describing the propagations
Output: void

```
1  if from_road == -1 then
2      for to_road in to_roads do
3          if to_road not in tree then
4              tree.add_node(to_road, route_id = to_road)
5              tree.add_edge(−1, to_road, freq = 0)
6          end
7          tree.edges[−1, to_road]['freq']+ = 1
8          current_propagations[to_road] = [to_road]
9      end
10 else
11     for c_prop in current_propagations[from_road] do
12         for to_road in to_roads do
13             prop_pointer = c_prop +' _' + to_road
14             if prop_pointer not in tree then
15                 tree.add_node(prop_pointer, route_id = to_road)
                   tree.add_edge(c_prop, prop_pointer, freq = 0)
16             end
17             tree.edges[c_prop, prop_pointer]['freq']+ = 1
18             current_propagations[to_road] .append(prop_pointer)
19             if to_road not in tree then
20                 tree.add_node(to_road, route_id = to_road)
21                 tree.add_edge(−1, to_road, freq = 0)
22             end
23             tree.edges[−1, to_road]['freq']+ = 1
24             if to_road not in current_propagations[to_road] then
25                 current_propagations[to_road] .append(to_road)
26             end
27         end
28     end
```

Algorithm 2: The pseudo code for $add\_propagations$ submethod

It may be that part of a propagation path occurs more frequently than the entire propagation path itself, e.g., from the propagation $R_1 \rightarrow R_2 \rightarrow R_3$, the propagation $R_2 \rightarrow R_3$ occurs more frequently. This may be due to the fact that a traffic jam has already occurred in R_2 or that R_2 is connected with another road segment R_4 from which the congestion often propagates as part of an $R_4 \rightarrow R_2 \rightarrow R_3$ propagation. To record these as well, we add the variable to_road to the graph describing the propagations ($tree$) as a congestion occurrence and also register this in the variable $current\_propagations$ to the road segment to_road (*lines 21-28*).

*B. Frequent Propagations (*FP*) method*

Input: $tree$: graph describing propagations $\mathcal{P}(\mathcal{V}, \mathcal{L})$, eps: ϵ frequency threshold
Output: $subtrees$: frequent propagation graphs

```
1  tree_cp = tree.copy()
2  removable_edges = []
3  for edge in tree_cp.edges do
4      if edge.freq < eps then
5          removable_edges.append(edge)
6      end
7  end
8  tree_cp.remove_edges_from(removable_edges)
9
10 isolated = get_isolated_nodes(tree_cp))
11 tree_cp.remove_nodes_from(isolated)
12
13 if not tree_cp.has_node(−1) then
14     return[]
15 end
16 propagation_sources = tree_cp.successors('−1')
17 sub_trees = []
18 for propagation_source in propagation_sources do
19     sub_tree = bfs_tree(tree_cp, propagation_source)
20     if sub_tree is not None then
21         sub_trees.append(sub_tree)
22     end
23 end
```

Algorithm 3: Pseudo code for the FP method

The task of the FP method is to find the frequent propagations based on the descriptive tree $\mathcal{P}(\mathcal{V}, \mathcal{L})$ constructed by the PT method and a ϵ threshold ($\epsilon \in \mathbb{N}^+$). To do this, the FP method iterate over the edges of the graph $\mathcal{P}(\mathcal{V}, \mathcal{L})$ and deletes all edges whose frequency $freq$ value is less than the threshold value. This will result in the originalϵ breaking into several subgraphs that now only contain edges that satisfy the condition.

It is worth noting that the graph $\mathcal{P}(\mathcal{V}, \mathcal{L})$ only needs to be calculated once, and after that any number of ϵ thresholds can be tested on it. This is a faster approach to the problem than if the propagation graph had to be rebuilt for each ϵ threshold.

The first step of the algorithm is to copy the original tree (*line 1*). This is necessary because the graph describing the propagation does not have to be recalculated for each new eps.

We then iterate over the edges of the copied tree and collect those with a $freq$ value less than eps (*lines 2-7*). The edges collected in the $removable\_edges$ variable are then deleted from the $tree\_cp$ tree (*line 8*).

After the step, there may be vertices in the graph that have no edges connected to them. Using the $get\_isolated\_nodes$ method, we collect these and then delete them from the graph as well (*lines 10-11*).

If there is no node -1 in the graph after that, it means that no edge met the condition, so we return with an empty array (*lines 13-15*).

In the last step, we go through the direct descendants of node -1, which are the starting road segments of the propagations (lines 16-23). Starting from each starting road segment ($propagation\_source$), we perform a breadth-first search with the bfs_tree method. In this, we traverse the subtree (sub_tree) and add it to the list of subtrees sub_tree), but only if the traversed subtree is not empty.

The number of steps of the FP method can be estimated by the cardinality of the edge sets ($|\boldsymbol{\mathcal{L}}|$) of the graph $\boldsymbol{\mathcal{P}}(\boldsymbol{\mathcal{V}}, \boldsymbol{\mathcal{L}})$. To delete edges that do not meet the ϵ threshold $|\boldsymbol{\mathcal{L}}|$ number of steps are required. The maximum number of steps of the breadth-first search even in the worst case can be estimated with the value of $|\boldsymbol{\mathcal{L}}|$. Thus the FP method's number of steps will be $\boldsymbol{O}(2|\boldsymbol{\mathcal{L}}|) \rightarrow \boldsymbol{O}(|\boldsymbol{\mathcal{L}}|)$. This also means that the number of steps of the entire SCPP algorithm is $\boldsymbol{O}(T|\boldsymbol{\mathcal{R}}| + |\boldsymbol{\mathcal{L}}|)$.

V. EVALUATION OF THE SCPP ALGORITHM

The performance of the SCPP algorithm was compared with the performance of the STC algorithm, because of all the methods described in the professional literature, only this one uses a quadratic number of steps.

In addition, the algorithm's author made the implementation of the method [48], and the test data set they used open source, so we could reproduce their results and compare them with the results of our own algorithm.

We were forced to modify the implementation of the STC algorithm at one point because we noticed that there was a theoretical error in the original implementation: The STCTree method of the algorithm [39] examines whether the traffic jam currently being examined may be a source of a previous propagation in a nested loop (line 20 of STCTree). However, when specifying the if condition, it does not take into account whether the previous propagation had already existed at the time of the occurrence of the currently examined traffic jam. Thus, it also included propagations in the comparison that did not actually exist.

In the first step of our study, we compared the output of the modified implementation of STC with the output of our own solution (Section V-A). Our goal was to examine the differences in the outputs of the two algorithms. We then wondered whether SCPP, which in theory is faster, would actually find frequent propagations sooner than STC, considering the changing input parameters (Section V-B).

In our performance analysis, we examined how much the average execution times depend on:

- the length of the examined time period,
- the size of the road network,
- the value of the threshold value.

The dataset [49] used in the evaluation for each test was the same and identical to that used in [39]. This is a real dataset recorded between June 17, 2013 and July 14, 2013 in Melbourne. The provided dataset contained only the required binary congestion data. Thus, we could not execute the Flow-speed ratio congestion definition on the dataset, but we could use it as an input for SCPP because it supports binary congestion data. The examined road network contains 586 road segments, from which data were collected every 5 minutes on average, and in total 7,657 times. Each test case was run at least 10 times to be sure that a temporary slow-down of the test system does not affect the results' correctness. During the evaluations, we used the average value of these executions' result.

A. Testing of the congestion propagation path identification

To be able to compare the outputs of the SCPP and STC algorithms, we need to better understand how the STC works. The STC algorithm uses the Apriori algorithm to discover frequent propagation paths (subtrees). The Apriori algorithm was basically invented to search for association rules (frequent coincidences) in a large database. A good example of this is understanding people's shopping habits, where the question is what products are purchased together by customers. With each purchase, the store saves what was in a customer's basket at the time of payment. On the database built from these baskets, we can execute the Apriori algorithm, which looks for products that were purchased together frequently. A product list will be common if the items in it are listed together at least ϵ times. This, in the context of a congestion study, means that a congestion path is considered to be frequent if the road segments within in the path have been congested together at least ϵ times.

It is important to note that STC and SCPP interpret the meaning of ϵ differently. While SCPP simply considers this value as a frequency ($\epsilon_{SCPP} \in \mathbb{R}^+$), the STC algorithm uses the Apriori algorithm to filter out frequent propagation, in which ϵ as a ratio to the total number of propagations found ($\epsilon_{STC} \in (0, 1)$) where the size of the set of propagations found is denoted by M. To make the results comparable, we first ran the STC procedure with the ϵ_{STC} values that are being examined. Based on this, a propagation is considered frequent if it has occurred at least $\epsilon_{STC} \times M$ times, so ϵ_{SCPP} can be calculated as $\epsilon_{SCPP} = \epsilon_{STC} \times M$. The SCPP algorithm was run with the ϵ_{SCPP} value derived from the STC algorithm, so the efficiency of the two methods can be compared.

To make the notation system simple in the following, when we refer to ϵ, we mean ϵ_{STC}.

To compare SCPP and STC, propagation paths were generated from the output propagation trees of both algorithms as shown in Figure 4 to include partial propagation paths in the comparison.

We then examined whether the propagation paths of one algorithm could be found among the propagation paths of the other algorithm. Performance-efficiency studies were run at different settings.

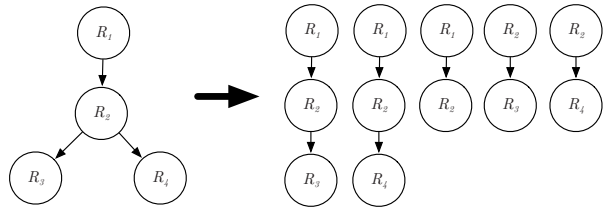


Fig. 4: Example of generating propagation paths.

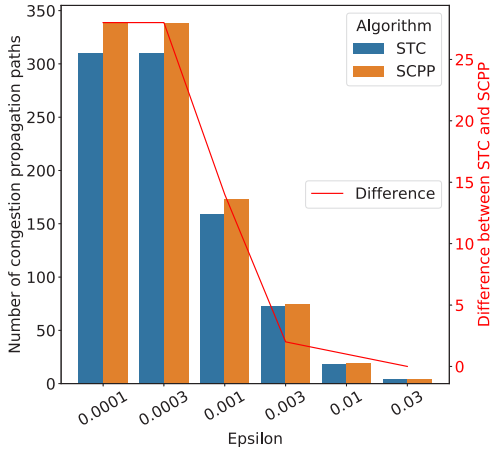


Fig. 5: Comparison of the number of traffic jam propagation paths found.

In all cases examined, it was true that the output of the SCPP included the output of the STC, but that STC did not include all the propagation paths of the SCPP. All of these were cases where the propagation path already consisted of at least 2 road segments and then propagated to a third road segment as the traffic jam on segment 2 disappeared. These cases were all successfully identified by the SCPP, while the STC was unable to do so. To compare the cardinality of the outputs, Figure 5 was prepared, where the number of propagation paths found was also displayed as a function of ϵ. At high ϵ values, the difference in the number of propagation paths found can be small, even 0. However, as we move toward the smaller ϵ values, the number of propagation paths not found by the STC begins to increase. In some cases, SCPP found up to 8.28% more, and on average 5.84% more, propagations than STC. This is because, in the case of less frequent propagations, the aforementioned situation when the STC is unable to identify the propagation path occurs more.

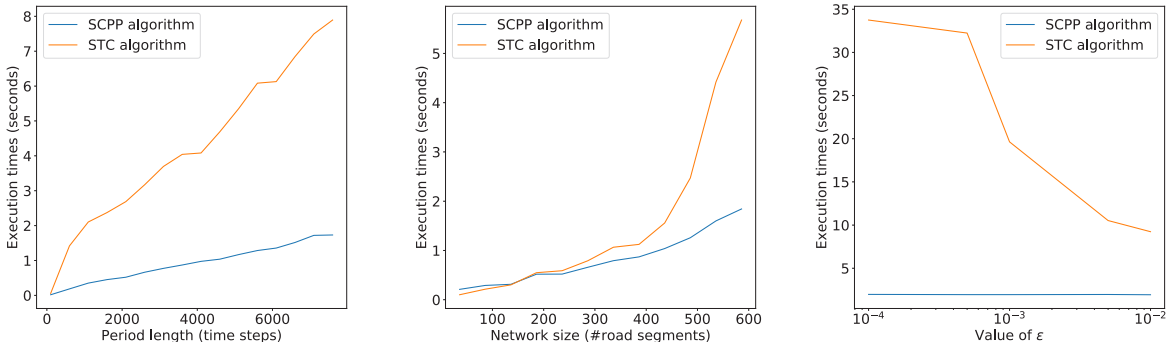
B. Study of computational performance

We started our measurements by changing the size of the studied time period. In these tests, we looked at how increasing the length of the studied time period (T parameter) changes the average execution time of the algorithms. Tests were started off with $T = 100$ and then increased step by step up to $T = 7657$. It can be seen on Figure 6a that with the increase of T, the average execution time of STC and SCPP both increase, but the STC is much steeper. The reason for this is that the STC's complexity is $\boldsymbol{O}(TN^2)$, while that of the SCPP is only $\boldsymbol{O}(TN)$, where T is the length of the examined time period in time intervals and N is the number of traffic jam phenomena that occurred during the period T. At maximum T, the STC execution time was 7.499 seconds, while the SCPP completed the task in 1.164 seconds. This means that SCPP solved the task 763% faster. Looking at all the studied cases, the SCPP performed calculations 483% faster on average.

We then looked at how increasing the number of road segments affect average execution times. First $|\boldsymbol{\mathcal{R}}| = 36$ road segments were examined, then we increased the number of road segments by 50 every time up to $|\boldsymbol{\mathcal{R}}| = 586$. The length of the studied time period was $T = 7657$ for all test cases. The results are shown in Figure 6b. It is clear that initially there is not much difference in the execution times of the two algorithms, but then the execution time of the STC starts to increase quadratically. The reason for this is that with the addition of new road segments, the number of traffic phenomena to be tested increases, of which the STC's complexity depends quadratically. Meanwhile, the runtime of the SCPP increases almost linearly. In the cases studied, SCPP was 146% faster on average, but at $|\boldsymbol{\mathcal{R}}| = 586$, the difference was 308%.

We were also curious about how much the size of the ϵ threshold value affects the average execution times. During the tests $|\boldsymbol{\mathcal{R}}| = 586$ road segments and $T = 7657$ time periods were used and only the value of ϵ was changed.

The result of the study is shown in Figure 6c, where the scaling of the x-axis is logarithmic. It can be seen that as the value of ϵ decreases, the average running time of the STC increases exponentially. This is because the Apriori algorithm has $\boldsymbol{O}(2^M)$ exponential complexity, where M is the number of total propagations found. As we decrease ϵ's value, the number of propagations that meet the frequency condition increase.

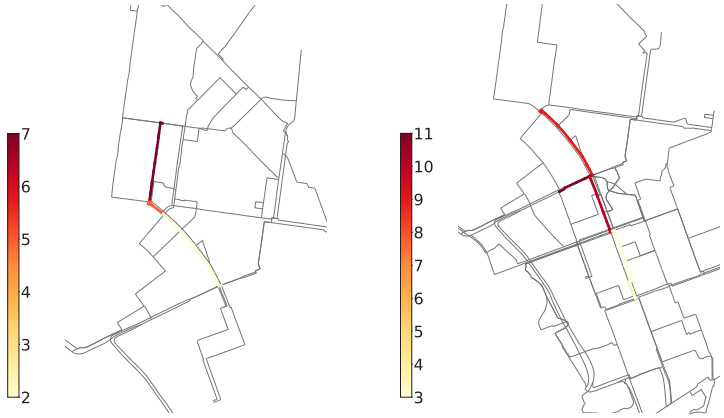


(a) The performance of the algorithms as a function of the studied time period

(b) The performance of the algorithms as a function of the number of road segments examined

(c) The performance of the algorithms as a function of ϵ

Fig. 6: Results of the performance study



(a) Example 1 (b) Example 2

Fig. 7: Visualizing the frequent propagations in Melbourne

In contrast, the average run time of the SCPP algorithm remained nearly constant, independent of ϵ's change. This is explained by the fact that within the SCPP algorithm, the step number of the Frequent Congestions (FC) method is $O(|\mathcal{L}|)$, which does not correlate with ϵ.

This resulted in the fact that in the studied cases the SCPP ran on average 10.79 times faster, while in the case of the lowest ϵ value examined the execution time was 17.13 faster.

C. Visualization of results

The SCPP algorithm not only provides the frequent propagation paths with greater accuracy and more quickly, unlike previous methods it also provides the quantification of propagation phenomena.

This is an extra piece of information when visualizing frequent propagations, so we can weight the frequency of propagation with appropriate coloring. The map used for visualization was downloaded from OpenStreetMap and the measurement points were mapped onto it. The propagation paths between the measurement points were determined by the Dijsktra algorithm running on the road network, which was weighted by the length of the road segments.

Figure 7 shows two different examples from the city of Melbourne. In the lower left corner of each example, there is a color scale showing the hues associated with the frequencies. With the visualization of SCPP, the propagation paths have become well identifiable, which in several cases branch out. In figure 7a spike-shaped propagations can be observed at two separate points. This is because in some cases, the GPS coordinates of the measurement points are not accurate, which distorts the output.

Visualization of traffic jams is important because urban traffic management authorities can discover correlations that can help them plan long-term urban transport (traffic light settings, public transport routes) and even shorter-term intervention.

VI. CONCLUSION

Managing the frequent congestion in the traffic networks of large cities is a serious challenge for municipal traffic managing organizations. In order to handle these situations, it is crucial to understand the processes that lead to congestion and propagation.

In this article, we introduce a new method capable of using city traffic data to find frequent traffic jam propagations. We introduce the steps of the method in detail and then we compare the output to the accepted and the widely cited solutions of the professional literature. In addition to introducing the method, we also lay out a new definition for "traffic jam" that, unlike previous solutions, does not rely on manually setting parameters, and instead is able to define traffic jam levels on the basis of the size of the road segment.

During our evaluation we look at how the performance of our method depends on the input parameters and real datasets. The results of our testing clearly show that SCPP carries out its task significantly faster and more precisely than the other solution, while also adding frequency information to the output, further aiding the refinement of the road network analysis and the visualization of propagations.

In the future, we would like to extend SCPP to use continuous congestion data instead of binary congestion data, as it can further improve the performance of the algorithm.

REFERENCES

[1] N. Zhong, J. Cao, and Y. Wang, "Traffic congestion, ambient air pollution, and health: Evidence from driving restrictions in beijing," *Journal of the Association of Environmental and Resource* Economists, vol. 4, no. 3, pp. 821–856, 2017, DOI: 10.1086/692115.

[2] M. Rosenlund, F. Forastiere, M. Stafoggia, D. Porta, M. Perucci, A. Ranzi, F. Nussio, and C. A. Perucci, "Comparison of regression models with land-use and emissions data to predict the spatial distribution of traffic-related air pollution in rome," *Journal of Exposure Science and Environmental Epidemiology*, vol. 18, no. 2, pp. 192–199, 2008, DOI: 10.1038/sj.jes.7500571.

[3] O. K. Kurt, J. Zhang, and K. E. Pinkerton, "Pulmonary health effects of air pollution," *Current opinion in pulmonary medicine*, vol. 22, no. 2, p. 138, 2016, DOI: 10.1097/MCP.0000000000000248.

[4] K. Chen, A. Schneider, J. Cyrys, K. Wolf, C. Meisinger, M. Heier, W. von Scheidt, B. Kuch, M. Pitz, A. Peters et al., "Hourly exposure to ultrafine particle metrics and the onset of myocardial infarction in augsburg, germany," *Environmental Health Perspectives*, vol. 128, no. 1, p. 017003, 2020, DOI: 10.1289/EHP5478.

[5] X. Tian, H. Dai, Y. Geng, J. Wilson, R. Wu, Y. Xie, and H. Hao, "Economic impacts from pm2. 5 pollution-related health effects in china's road transport sector: A provincial-level analysis," *Environment international*, vol. 115, pp. 220–229, 2018, DOI: 10.1016/j.envint.2018.03.030.

[6] L. Zhu, F. R. Yu, Y. Wang, B. Ning, and T. Tang, "Big data analytics in intelligent transportation systems: A survey," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 1, pp. 383–398, 2018, DOI: 10.1109/TITS.2018.2815678.

[7] J. Li, Y. Zhang, and Y. Chen, "A self-adaptive traffic light control system based on speed of vehicles," in *2016 IEEE International Conference on Software Quality, Reliability and Security Companion (QRS-C)*. IEEE, 2016, pp. 382–388, DOI: 10.1109/QRS-C.2016.58.

[8] L. A. Klein, M. K. Mills, and D. R. Gibson, "Traffic detector handbook: 3rd ed. — volume II," Federal Highway Administration, Tech. Rep. Publi. No. FHWA-HRT-06-139, October 2006.

[9] Z. MacHardy, A. Khan, K. Obana, and S. Iwashina, "V2x access technologies: Regulation, research, and remaining challenges," *IEEE Communications Surveys & Tutorials*, vol. 20, no. 3, pp. 1858–1877, 2018, DOI: 10.1109/COMST.2018.2808444.

[10] A. Paranjothi, M. S. Khan, and S. Zeadally, "A survey on congestion detection and control in connected vehicles," *Ad Hoc Networks*, vol. 108, p. 102277, 2020, DOI: 10.1016/j.adhoc.2020.102277.

[11] T. Afrin and N. Yodo, "A survey of road traffic congestion measures towards a sustainable and resilient transportation system," *Sustainability*, vol. 12, no. 11, p. 4660, 2020, DOI: 10.3390/su12114660.

[12] S. Wang, X. Zhang, J. Cao, L. He, L. Stenneth, P. S. Yu, Z. Li, and Z. Huang, "Computing urban traffic congestions by incorporating sparse gps probe data and social media data," *ACM Transactions on Information Systems (TOIS)*, vol. 35, no. 4, p. 40, 2017, DOI: 10.1145/3057281.

[13] S. Wang, F. Li, L. Stenneth, and S. Y. Philip, "Enhancing traffic congestion estimation with social media by coupled hidden markov model," in *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*. Springer, 2016, pp. 247–264, DOI: 10.1007/978-3-319-46227-1_16.

[14] Y. Yang, Y. Xu, J. Han, E. Wang, W. Chen, and L. Yue, "Efficient traffic congestion estimation using multiple spatio-temporal properties," *Neurocomputing*, vol. 267, pp. 344–353, 2017, DOI: 10.1016/j.neucom.2017.06.017.

[15] X. Kong, Z. Xu, G. Shen, J.Wang, Q. Yang, and B. Zhang, "Urban traffic congestion estimation and prediction based on floating car trajectory data," *Future Generation Computer Systems*, vol. 61, pp. 97–107, 2016, DOI: 10.1016/j.future.2015.11.013.

[16] S. Yang, "On feature selection for traffic congestion prediction," *Transportation Research Part C: Emerging Technologies*, vol. 26, pp. 160–169, 2013, DOI: 10.1016/j.trc.2012.08.005.

[17] M. Fouladgar, M. Parchami, R. Elmasri, and A. Ghaderi, "Scalable deep traffic flow neural networks for urban traffic congestion prediction," in *2017 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 2017, pp. 2251–2258, DOI: 10.1109/IJCNN.2017.7966128.

[18] M. Chen, X. Yu, and Y. Liu, "Pcnn: Deep convolutional networks for short-term traffic congestion prediction," *IEEE Transactions on Intelligent Transportation Systems*, vol. 19, no. 11, pp. 3550–3559, 2018, DOI: 10.1109/TITS.2018.2835523.

[19] Y. Wang, J. Cao, W. Li, and T. Gu, "Mining traffic congestion correlation between road segments on gps trajectories," in *2016 IEEE International Conference on Smart Computing (SMARTCOMP)*. IEEE, 2016, pp. 1–8, DOI: 10.1109/SMARTCOMP.2016.7501704.

[20] H. Xiong, A. Vahedian, X. Zhou, Y. Li, and J. Luo, "Predicting traffic congestion propagation patterns: A propagation graph approach." in *IWCTS@ SIGSPATIAL*, 2018, pp. 60–69, DOI: 10.1145/3283207.3283213.

[21] M. Bai, Y. Lin, M. Ma, P. Wang, and L. Duan, "Prepct: Traffic congestion prediction in smart cities with relative position congestion tensor," *Neurocomputing*, 2020, DOI: 10.1016/j.neucom.2020.08.075.

[22] N. H. Gartner, C. J. Messer, and A. Rathi, "Traffic flow theory-a state-of the-art report: revisfed monograph on traffic flow theory," *FHWA, U.S. Department of Transportation*, 2002.

[23] A. M. Rao and K. R. Rao, "Measuring urban traffic congestion-a review." *International Journal for Traffic & Transport Engineering*, vol. 2, no. 4, 2012, DOI: 10.7708/IJTTE.2012.2(4).01

[24] F. He, X. Yan, Y. Liu, and L. Ma, "A traffic congestion assessment method for urban road networks based on speed performance index," *Procedia engineering*, vol. 137, pp. 425–433, 2016, DOI: 10.1016/j.proeng.2016.01.277.

[25] D. Ter Huurne and J. Andersen, "A quantitative measure of congestion in stellenbosch using probe data," in *Proc. of the 1st Int. Conf. on the use of Mobile Informations and Communication Technology (ICT) in Africa UMICT*. STIAS Conference Centre, 9-10 Dec 2014, Stellenbosch, South Africa.

[26] M. Aftabuzzaman, "Measuring traffic congestion-a critical review," in *30th Australasian Transport Research Forum*, 2007, pp. 1–16.

[27] C. Wan, Z. Yang, D. Zhang, X. Yan, and S. Fan, "Resilience in transportation systems: a systematic review and future directions," *Transport reviews*, vol. 38, no. 4, pp. 479–498, 2018, DOI: 10.1080/01441647.2017.1383532.

[28] J. Tang and H. R. Heinimann, "A resilience-oriented approach for quantitatively assessing recurrent spatial-temporal congestion on urban roads," *PLoS one*, vol. 13, no. 1, p. e0190616, 2018, DOI: 10.1371/journal.pone.0190616.

[29] *Highway Capacity Manual 6th Edition: A Guide for Multimodal Mobility Analysis*. TRB, 2016.

[30] X. Di, Y. Xiao, C. Zhu, Y. Deng, Q. Zhao, and W. Rao, "Traffic congestion prediction by spatiotemporal propagation patterns," in *2019 20th IEEE International Conference on Mobile Data Management (MDM)*. IEEE, 2019, pp. 298–303, DOI: 10.1145/3283207.3283213.

[31] S. Xingjian, Z. Chen, H. Wang, D.-Y. Yeung, W.-K. Wong, and W.-c. Woo, "Convolutional lstm network: A machine learning approach for precipitation nowcasting," in *Advances in neural information processing systems*, 2015, pp. 802–810.

[32] Y. Liang, Z. Jiang, and Y. Zheng, "Inferring traffic cascading patterns," in *Proceedings of the 25th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems*. ACM, 2017, p. 2, DOI: 10.1145/3139958.3139960.

[33] A.-L. Barabasi, "The origin of bursts and heavy tails in human dynamics," *Nature*, vol. 435, no. 7039, p. 207, 2005, DOI: 10.1038/nature03459.

[34] M. G. Rodriguez and B. Schölkopf, "Submodular inference of diffusion networks from multiple trees," *arXiv preprint arXiv:1205.1671*, 2012.

[35] R. Albert and A.-L. Barabási, "Statistical mechanics of complex networks," *Reviews of modern physics*, vol. 74, no. 1, p. 47, 2002, DOI: 10.1103/RevModPhys.74.47.

[36] W. Liu, Y. Zheng, S. Chawla, J. Yuan, and X. Xing, "Discovering spatiotemporal causal interactions in traffic data streams," in *Proceedings of the 17th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2011, pp. 1010–1018, DOI: 10.1145/2020408.2020571.

[37] L. Di Stefano and A. Bulgarelli, "A simple and efficient connected components labeling algorithm," in *Proceedings 10th International Conference on Image Analysis and Processing*. IEEE, 1999, pp. 322–327, DOI: 10.1109/ICIAP.1999.797615.

[38] B. Hossain, K. A. Adnan, M. F. Rabbi, and M. E. Ali, "Modelling road traffic congestion from trajectories," in *Proceedings of the 3rd International Conference on Data Science and Information Technology*, 2020, pp. 117–122, DOI: 10.1145/3414274.3414491.

[39] H. Nguyen, W. Liu, and F. Chen, "Discovering congestion propagation patterns in spatio-temporal traffic data," *IEEE Transactions on Big Data*, vol. 3, no. 2, pp. 169–180, 2016, DOI: 10.1109/TBDATA.2016.2587669.

[40] R. Agrawal, R. Srikant et al., "Fast algorithms for mining association rules," in *Proc. 20th int. conf. very large data bases, VLDB*, vol. 1215, 1994, pp. 487–499,

[41] Z. Chen, Y. Yang, L. Huang, E. Wang, and D. Li, "Discovering urban traffic congestion propagation patterns with taxi trajectory data," *IEEE Access*, vol. 6, pp. 69 481–69 491, 2018, DOI: 10.1109/ACCESS.2018.2881039.

[42] J. Han, J. Pei, and Y. Yin, "Mining frequent patterns without candidate generation," in *ACM sigmod record*, vol. 29, no. 2. ACM, 2000, pp. 1–12, DOI: 10.1145/335191.335372.

[43] L. A. Elefteriadou, "New trb publication: The highway capacity manual: A guide for multimodal mobility analysis," *TR News*, no. 306, 2016.

[44] B. Friedrich, "The effect of autonomous vehicles on traffic," in *Autonomous Driving*. Springer, 2016, pp. 317–334, DOI: 10.1007/978-3-662-48847-8_16.

[45] Q. Lu, T. Tettamanti, D. Hörcher, and I. Varga, "The impact of autonomous vehicles on urban traffic network capacity: an experimental analysis by microscopic traffic simulation," *Transportation Letters*, vol. 12, no. 8, pp. 540–549, 2020, DOI: 10.1080/19427867.2019.1662561.

[46] A. Ghiasi, "Connected autonomous vehicles: Capacity analysis, trajectory optimization, and speed harmonization," Ph.D. dissertation, University of South Florida, 2018.

[47] "Caltrans pems," 2020, [Online; accessed 17-March-2020]. [Online]. Available: http://pems.dot.ca.gov/

[48] H. Nguyen, "congestionpropagation," https://github.com/nguyend/congestionpropagation, 2021, [Online; accessed 22-March-2021].

[49] ——, "Melbourne congestion dataset," https://github.com/nguyend/congestionpropagation/tree/master/matdata, 2021, [Online; accessed 22-March-2021].

Attila M. Nagy received the B.S. and M.S. degrees in computer engineering from the Budapest University of Technology and Economics (BME), Budapest, in 2016. From 2016-2020, he was PhD student in computer engineering from the Budapest University of Technology and Economics (BME), Budapest. Since 2020 he is a Research Assistant in MEDIANETS laboratory at Department of Networked Systems and Services, BME. His research interests include time series data mining and analysis, traffic prediction, traffic congestion data analysis, automatic traffic incident detection.

Vilmos Simon received his PhD from the Budapest University of Technology and Economics (BME) in 2009. Currently he is an Associate Professor at the Department of Networked Systems and Services, Head of the Multimedia Networks and Services Laboratory and Deputy Head of Department of Networked Systems and Services.

He has done research on mobility management and energy efficiency in mobile cellular systems and self-organized mobile networks, recently his research interests include machine learning and data analytics for smart cities and intelligent transportation management systems.

He published 50+ papers in international journals and conferences, and acts as a reviewer or organizer for numerous scientific conferences. He serves currently as the Corporate liaison vice president for the Connected and Automated Mobility Cluster of Zala.