Change point detection in visualizing bus link travel times

Mykola Andrushchenko

University of Tampere
Faculty of Natural Sciences
Degree Programme in Software Development
M. Sc. thesis
Supervisor: Jyrki Nummenmaa
December 2017

University of Tampere
Faculty of Natural Sciences
Degree Programme in Software Development
Mykola Andrushchenko: Change point detection in visualizing bus link travel times
M. Sc. thesis, 44 pages, 2 index pages
December 2017

The volume and availability of traffic data has grown tremendously in recent years, and applications based on their analysis continue to offer more and more services to the general public. This work presents a practical implementation of the concept of bus links, i.e. bus stop pairs, an approach that has mostly been discussed theoretically up to now. The requirements and scope of the development have been influenced by findings attained in prior work.

The implementation represents an online tool used to visualize the current traffic situation in the city of Tampere by colouring bus links in accordance with their estimated congestion. Data packets utilized in the visualization can be also dispatched to other clients for further analysis. Consequently, the tool is able to provide information relevant to commuters and data analysts while avoiding the computational load typically associated with processing large volumes of input data.

The problem of change point detection arose in the analysis of link data. The traffic model recording the relative fluency of each link was required to store indicator values for distinct periods of traffic flow, and thus identify their endpoints from the respective link's travel times. The implementation was adjusted to perform a number of tests in order to select the most suitable algorithms for this task, with a practical emphasis on their performance in the given application. Change point distributions produced by various methods were considered in terms of their feasibility, both in the visualization and in possible data analysis.

Key words and terms: traffic data, public transport, bus link, change point.

Contents

Acknowledgements

I am immensely grateful to my supervisor, Professor Jyrki Nummenmaa, for his support throughout the entire thesis process, from proposing the initial topic all the way to the final draft. His incisive comments have consistently steered this work towards a finer state, whether scientifically, structurally or stylistically.

I am also indebted to Paula Syrjärinne for her assistance in the early stages of the work. The data files and components already assembled by her were invaluable in the development process, accelerating various side tasks significantly. Above all, her patient explanations eventually revealed the scope and the functionality of the prospective tool to me.

1. Introduction

Collection and analysis of traffic data have become remarkably popular in recent years. Statistical recommendations derived from a sufficiently large dataset can greatly improve the traffic situation in a particular location at a low cost, and may even serve to introduce changes in traffic infrastructure [Hill, 2014]. A significant effort in this field has been applied in many of the world's cities, and quite often it has resulted in active cooperation: city administrations work hard to maintain a steady source of traffic data, typically made publicly available, while researchers may provide feedback based on the conclusions they infer from the data [Zambrano et al., 2016].

There are many potential methods of obtaining traffic data. When the public transport network is sufficiently developed, a lot of information can be acquired by equipping individual vehicles with GPS-based transmitters [Baptista et al., 2012]. While navigating their established routes, they periodically broadcast messages containing their current location, speed, delay compared to the timetable, etc. These messages are aggregated and possibly grouped for further analysis. A more versatile option is represented by traffic light data, commonly gathered from sensitive detectors installed at road intersections [Ye et al., 2016]. Such detectors can identify when and how long the intersection is occupied by vehicles, although the type of the vehicle in question may not be easily identifiable. More exotic data sources, such as crowdsourcing applications, can also be considered [Lau and Ismail, 2015].

A recent development in the field involves the use of bus links, i.e. pairs of bus stops. Instead of examining individual vehicles, this approach observes the parameters of particular bus stop pairs, measuring the average time it takes a bus to navigate between them. While a number of papers have already discussed the subject in some detail or dealt with issues closely related to the approach, the new concept still lacks a comprehensive practical implementation. Certain lesser concerns, such as required data formats, likewise remain unresolved.

Generally, the observed travel times can be accumulated to form a statistical model of the traffic situation. The average travel times for each link, as well as other auxiliary values, are recorded for every distinct period in a given timeframe, most commonly a single day. This model is then used as a reference to determine the link states at a later point: by comparing the current travel times to the values found in the model, a conclusion can be reached about the present state of each link. The rules followed in reaching such decisions may be greatly varied according to the size of the observed environment, typical traffic intensity expected in it, or the nature of irregularities in the traffic flow.

Significant changes in the public transport system, such as the introduction of new timetables or new lines of transportation, will likely require the model to be updated or even recreated from scratch. The latter is associated with a period of inaccurate predictions until the model accumulates a large set of observations, at which point it generally begins to reflect the new traffic circumstances correctly. Nonetheless, it is also essential to make small updates to the current model in the absence of such radical changes. The actual traffic flow is assumed to be similar to the most recent observations, those recorded in the last few hours or days, and these travel times should therefore have a limited impact on the model.

To be precise, the model must be aware of the “distinct periods” mentioned above. A single value to characterize daily travel times for a particular link is certainly too coarse a description. Instead, the model must retain the average values for every period of time distinguished by its traffic conditions, and thus identify such periods using nothing but the travel times themselves. This is known as the problem of change point detection.

Originally a statistical issue, locating change points thus becomes a necessity during the analysis of link travel times. Instances when the observations change their character are likely to delimit distinct phases of traffic flow: generally, travel times can be expected to increase or decrease, which corresponds directly to more or less intense traffic. Many algorithms have been developed for the purpose of finding change points in a sample of observations. However, it is not immediately obvious which of them are suitable in the current context, for both theoretical and pragmatic reasons. For instance, a rigorous procedure may still be unsuitable if it produces too many or too few change points, so that the resulting traffic states are no longer of practical use.

The benefits of the bus link model are seen both in its realization and potential clients. From the implementation perspective, it offers a significant reduction in computational cost: focusing on the bus stops themselves permits the application to discard most of the information sent in transit between them, which in turn lessens processing times and storage costs. The sheer amount of input data, especially in areas with well-developed transport infrastructures, can require complex and often costly solutions. The link model is an attempt to avoid this complexity while still retaining a representative view of the data.

From the viewpoint of a prospective user, the model may seem rather uninspiring: for every link, it merely provides a set of reference values that the current travel times can be compared to. The model can also provide its own logic that assigns the state of the link out of several possible options, but there are reasons to leave that decision to the user.

However, these reference values may in fact be utilized in many ways. The simplest of them is perhaps visualization: by comparing the actual travel times with the model’s values for all its links, an overall image of traffic congestion in the area can be immediately formed. As long as the model also retains older values, a historical perspective of every particular street and intersection will also be available. Furthermore, the bus link model can be similarly used to test proposed changes in the public transport system, such as a rearrangement of traffic lights in a given district. By examining how the values stored by the model have evolved after the change, or how the current travel times are rated according to these values, the efficiency of the change can be evaluated with ease.

Moreover, the existence of a "standard" travel time defined by the link model naturally leads to the appearance of exceptional travel times, in particular unusually high ones. Knowing that a specific link is congested does not lead to a direct conclusion about the cause of the delays, but provides a starting point for the investigation. For instance, the regularity of the congestion period may decide whether it is a persistent occurrence, such as road maintenance or a traffic jam.

With an awareness of the concerns listed above, this thesis documents the functionality, constraints and properties of a "traffic awareness" tool, which consists of a data collector component, a traffic model generator and a small visualization package. The package makes use of the model to display the current bus link states in the city of Tampere, superimposed onto a map in real time. However, the model may be freely used by other client applications as well. Notably, the processing time and disk space requirements are fairly modest given the initial volume of input data. The development of the tool has addressed and successfully resolved the following research questions:

- How can we define the boundary between normal traffic, peak traffic and other exceptional states?
- Where do changes occur in a recorded sequence of travel times?
- How is the established model influenced and adjusted by the most recent data?

Prior work related both to bus links and to change point detection is explored more extensively in the following section. The tool itself is documented in Section 3, while Section 4 is dedicated to a number of experiments aimed at choosing appropriate change point detection algorithms. Several methods are tested with different parameters to determine which ones are particularly suitable in the current setting. The thesis concludes with a summary of the results achieved and a few suggestions for further development.

2. Literature review

2.1. The link model

2.1.1. Motivation and basic concepts

Public transport data represent a rather broad source of information in modern research. Different cities have come forward with their own implementations of the service, although there are a few common features. Typically, the data are distributed according to an established specification, GTFS being a prime example [GTFS], and made available through a separate API. A similar approach has been taken in the city of Tampere. Bus data are provided in the form of a JSON document, updated once every second to reflect the latest messages dispatched by the vehicles [J_API]. This document contains information about each vehicle's state, including its identifier, line number, direction, speed, coordinates and possible lateness. If these details are properly stored and analyzed, they can serve as a foundation for many practical applications. In particular, a route planner predicting the likelihood of a successful bus connection has already been developed and deployed.

Still, this format is also associated with notable difficulties in storage and processing. The original traffic data, as obtained from public APIs, are often rather verbose and require a lot of storage space. Months of observation data may be required to obtain a truthful statistical representation of certain processes, which proves rather challenging in practice. For instance, typical SQL-based solutions begin to lose performance in storage and retrieval tasks when the number of database entries goes into the millions (however, this can be compensated with certain maintenance techniques or hardware enhancements). Moreover, a system dealing with real-time traffic conditions is expected to provide its output within minutes or even seconds; this performance may be only achievable with computing clusters, which further complicates the development, testing and deployment processes. It would thus be fitting to consider another data format, one that omits a significant portion of the available information while retaining the most practically relevant of its characteristics.

The approach to be taken in this thesis work relies extensively on the concept of links, as formulated by Syrjärinne and Nummenmaa [2016]. A link is defined as the road segment between two consecutive bus stops, i.e. two stops passed directly one after another by at least one bus line. The same pair of stops forms a different link when taken in the other direction, and is physically located on the opposite side of the road. The entire public transport network of a city can then be broken down into a number of links. Since every pair of nearby bus stops can form a link, the total number of links can be quite high even for relatively small cities, although most of them will be used rather infrequently.

The word "link" is not a new term in the field of transport. However, this thesis and prior work in the same direction has employed this word in a somewhat different sense than other sources. For instance, Shimizu et al. [2010] characterize links as road segments, with respect to intersections and traffic lights rather than bus stops, and the resulting object is similar to the one defined here. Analogous usage of the term is apparently found in Liang and Ling-xiang [2006]. Likewise, the compound term "link travel time", as examined by Jones et al. [2013] or Dong et al. [2012], refers to the time it takes a vehicle to navigate a link, with the definition of the link itself slightly different from the one used here.

The primary advantage of using links lies in the data compression opportunities they provide. The link-based approach enables the recording of observations related to the arrivals and departures pertinent to a particular bus stop, while all other entries are discarded (see Figure 1). This may seem a loss of significant information, but a wide range of applications can still function on the basis of this limited dataset. The resulting data volume is smaller than the original one by a factor of about 30, which compensates even for a high number of links.

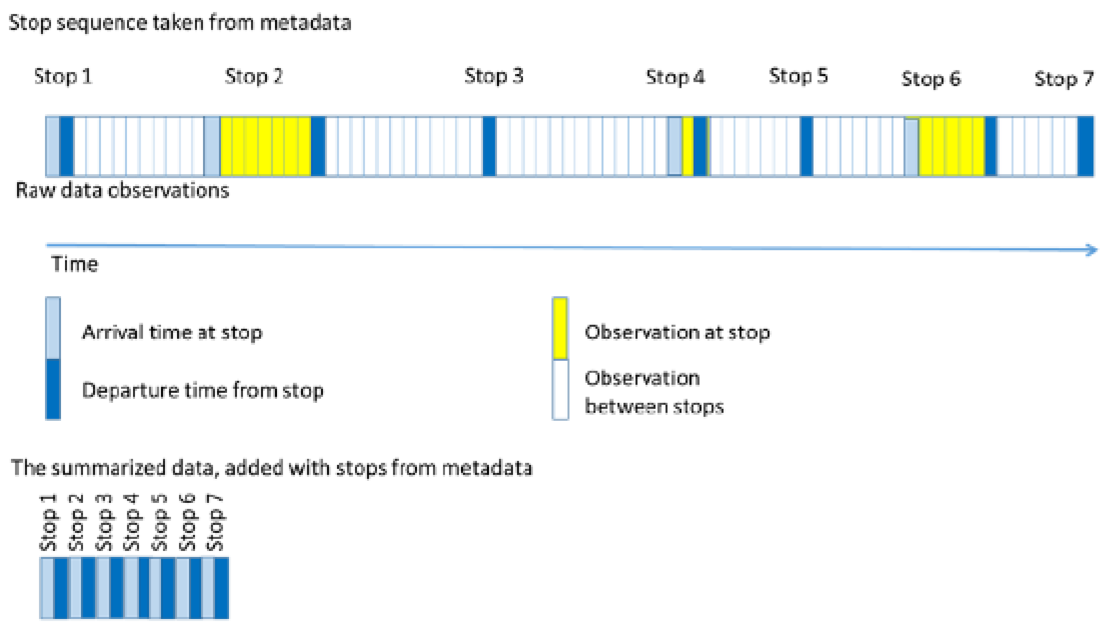

Figure 1. Traffic data compression [Syrjärinne and Nummenmaa, 2015]

This reduction also makes the proposed concept undemanding in terms of input formats. While various systems can provide traffic data in many distinct forms, hopefully standardized to some extent, the link model requires only the most basic fields for its functioning: the vehicle's coordinates, the timestamp of the message, and the static coordinates of the bus stops utilized. (The Journeys API used in Tampere also provides the codes of the next and the previous stop for every active bus, but these details may be derived from different sources.) Other values, including the speed of the bus, its direction and schedule, are not relevant to the model.

2.1.2. Data reliability

Despite the relatively rigorous specification of the traffic data format, the messages arriving from a data source (the “raw data observations” in Figure 1) cannot be “blindly” recorded and processed. Inaccuracies in the original data can emerge from many sources, and it is vital to remove them while preprocessing data for the link model. These inaccuracies include the following:

1) Incorrect sensor readings. Occasionally, a bus continues to transmit data even when it has finished its route, or when it is in transit towards its scheduled route without taking any passengers. It is also possible for a bus to send messages labelled with another vehicle’s identifier, which may then appear to originate from a different part of the city altogether. Such messages are typically outliers, and in practice they are merely removed from the original data. On the other hand, some bus transmissions simply stop arriving for brief intervals of time. This means that intermediate bus positions during the “silent” period can only be determined through interpolation.

2) Inaccurate bus stop detection. While the coordinates of every bus stop are known in advance, it is not always possible to identify precisely when a bus approaches or leaves one. The vehicle’s sensors introduce a certain error into the measurements. The behaviour of a bus at every given stop is not fully predictable, either: a bus may ignore a stop completely if no passengers wish to board the bus or leave it, or it may be forced to stop some distance away from the actual stop. This is a common occurrence when another bus is already occupying the stop, but other factors, such as roadwork, can also lead to this situation. In practice, a certain radius is assigned to every bus stop, and vehicles that enter the circle defined by the radius are considered to have reached the stop. The exact value of the radius can only be determined practically. If it is too large, measurements will be affected by external factors: for instance, a bus stopping at a traffic light sufficiently close to a stop would be assumed to have already arrived at the stop. Conversely, a radius too small would run the risk of losing actual measurements. It increases the probability of a vehicle passing through the whole circle without sending a single message, so that the receiver would have to somehow deduce the presence of the bus stop “between” the coordinates in the next and previous messages. A bus can also be fairly close to a bus stop in terms of physical distance but far in terms of road distance. This may occur since geographical coordinates do not account for the possible curvature of the road segment, nor for the case of multiple overlapping roads intersecting in the same area.

3) Data update interval. The API used in practice refreshes the data once per second. While this is generally sufficient for most practical purposes, the “reduced” format of the link model can lead to inaccurate conclusions. Given the radius concept above, if a bus sends the current measurement just outside a bus stop’s range and the next one well inside it, the

discrepancy between the actual and the perceived arrival time can be quite significant. In this case, it scales with the speed of the bus at the time of its arrival.

These issues have been originally encountered in the practical setting of data collection for the tool described in this thesis. Obviously, many other hindrances may occur in other areas; these particular ones are listed merely as examples of possible complications. Still, they are mentioned to some extent in existing literature. For instance, an evaluation of London's bus priority system, aided by traffic lights, covers the aspect of bus stop detection [Hounsell et al., 2012]. The link model's robustness and its dependency on the interval between the original observations are discussed by Mu et al. [2010]. From another perspective, quality in terms of data regularity is elaborated upon by Barabino et al. [2013]. Some thoughts on data denoising, together with a different procedure for bus stop detection, can be found in Pinelli et al. [2015].

Ultimately, these studies are often concerned with the irregularity or insufficiency of the traffic data under consideration. In the context of the present work, however, data availability is not a concern. Rather, the occasional uncertainties in the data complicate the identification and detection of various conditions necessary for the operation of the link model. In spite of the aforementioned difficulties, it is still feasible to remove the majority of the inconsistent data at the preprocessing stage, leaving the model to operate on a large share of the original data. The contribution of the present work does not extend to the preprocessing methods themselves: instead, the newly developed tool was already able to regard its input observations as reliable, without filtering them further.

2.1.3. Filling and updating the link model

Before explaining how the raw traffic data are shaped into the bus link model, it is worth briefly looking at the two principal "styles" of potential applications using this model. One of them is the offline mode, which involves a fixed set of observations accumulated, for instance, over the course of one day. This is generally convenient since the data can be freely sorted and filtered. However, it may be also impractical for applications that provide real-time assessment of the traffic situation.

When the application in question can only operate on data received up to the present moment, the online mode must be used. It can be referred to as an instance of streaming, since the newly dispatched data must be picked up by the processing application almost instantly. This particular demand is adequately handled by existing services. In the Hadoop infrastructure, for instance, Kafka and Flume are two services commonly employed to arrange data delivery. They have been designed to be easily extendable to multiple receivers, senders and intermediate agents; they also scale well to large volumes of data [Kafka].

Apart from a different data source, online processing also requires some changes from an algorithmic perspective. Since observations are not necessarily associated with trips by default, the application must determine whether to assign new messages to currently monitored trips or create new trips instead. It is equally important to detect when a particular bus has run its course and should not be sending any more messages, which should cause its trip to be “closed”. More generally, the problem of preprocessing covered in the previous subsection is also applicable to streaming data. It represents an even greater concern in this case, as the preprocessing must not cause the application to fall behind the data stream.

In either case, the link model is based on the observations collected from a traffic data API, with the necessary preprocessing performed and with messages occurring between bus stops discarded. This should relax the memory requirements of the tool and remove at least some issues often encountered in streaming applications.

The most important variable gathered from the raw observations is undoubtedly the travel time, i.e. the time taken by the bus to travel between the two bus stops forming the endpoints of a link. This is effectively the difference between the timestamps of the messages corresponding to a vehicle’s arrival and departure from the respective bus stops. The reduction in the input volume, and accordingly in the computational cost, is mostly due to the fact that bus messages are delivered at a high rate, perhaps once per second, but the ones related to bus stops are far less frequent. It should also be noted that the system does not depend on particular bus lines or schedules: the observations will be recorded properly as long as the same bus stops are used and the same GPS messages are broadcast by vehicles moving between them.

Given a sufficiently long monitoring period, the average travel time over every link can be determined with respect to the entire day or a particular time of the day. However, a more robust approach involves choosing the median and possibly employing other quantiles in further analysis as well. The inconsistencies mentioned above often produce unnaturally high or low travel times, thus distorting the mean value, while the median is relatively resistant to this. By encoding every link with its two endpoints, we arrive at the following brief representation shown in Table 1:

	Interval 1	Interval 2	…	Interval M
Link 1	45.5	45.5	…	38.0
Link 2	22.1	21.0	…	21.0
…	…	…	…	…
Link N	…	…	…	…

Table 1. A possible format of the link model

The actual number of intervals represented by the table’s columns can be determined in different ways, since their length may well be variable. Assuming that Table 1 contains median travel times for every interval, a similar table can be constructed for a different statistic. The present work makes use of both the median and the 90th percentile.

Whenever the current travel time is sufficiently greater than the “average”, most likely represented by the median, it can be regarded as a potential issue with the link. Two likely causes of such issues are traffic jams and road accidents: both slow down all traffic, including buses, within a restricted road segment. While the details retained in the link model are not sufficient by themselves to determine the exact reason for the delay, it can be said that traffic jams are often persistent, occurring repeatedly at roughly the same time, while accidents are singular occasions.

Travel times cannot be compared between links, since each link differs in its layout and length. However, a reasonable comparison can be drawn between the levels observed at different links. A level is a function based on the ratio between the median travel time observed in a particular interval and the overall median travel time. While this can be expressed in different scales, high levels generally correspond to more intense traffic while low levels indicate traffic with below-average intensity [Syrjärinne, 2016].

Whereas the previous discussion has focused on link models constructed on the basis of a single day, it is in fact appropriate to utilize a larger volume of historical data when composing the model. This naturally adds reliability to any statistical properties of the resulting product. Such an extension is technically simple: the methods used to classify observations can still be applied across multiple days without any changes. However, the relevance and accuracy of such a model are brought into question.

First of all, it is possible that a longer observation period includes data of principally distinct kinds. For example, when different timetables are used for different seasons of the year, any period spanning two seasons would bring together two distinct sets of travel times. Thus, generalizations derived from this union would not be adequate for either set. Discrepancies in travel times can also arise from natural conditions, such as periods of intense rain and snow. This effectively limits the maximum length of the timeframe used to build the model. A similar concern emerges when different timetables are followed on weekdays and weekends, although this can be handled by constructing separate models for each timetable involved.

Secondly, there are many ways to consider individual observations within a homogeneous period. The simplest approach would be to treat every day of the period in question the same, but this might not be the most accurate way. It could be argued that the most recent observations would be more indicative of the future situation than somewhat older ones, and as such should be given more weight in the model.

However, there are many ways to emphasize certain observations. One approach is to update the model using a clustering algorithm, for instance, on a daily basis, as proposed by Hu and Hellendoorn [2013]. It is almost certain that a regular process, such as a daily or a weekly update, will be necessary to incorporate the most recent changes. In any case, the example of Table 1 will most likely become a “daily model” of sorts, while the “link model” itself will be represented by a composition of multiple daily models. A possible implementation of this idea is covered in Section 3.2.

2.1.4. Applications

Although the application developed within this thesis is primarily a visualization tool, used to depict the link states in real time, the concept allows for other extensions as well. The same mechanism can be used to view historical data related to any particular link. As long as those travel times are retained, it is possible to reconstruct the link model for the corresponding period with relative ease. It may also be instructive to observe individual links over time, superimposing their travel times on the same plot or aggregating them in a single table.

More generally, the statistics provided by the link model can serve as indicators of potential issues with the public transportation system. If the non-fluent states of each link are duly noted, it is feasible to determine particularly congested links over a given timeframe, e.g. at specific times of the day. Syrjärinne et al. [2015] offer a fine example of the analysis that can be carried out in this manner. While the model cannot adequately explain the causes of congestion, determining these should be simple enough with the exact times already identified.

As was mentioned above, exceptionally high travel times observed at a link can be viewed as a potential issue occurring in the vicinity of that link. A tool that employs the link model can thus be used for the purpose of incident detection. In this task, the system has to raise an alarm once it detects a sequence of abnormally high travel times at a given link. The alarm is therefore produced some time after the event. A coefficient can be used to determine this timeframe, and the value of this coefficient must be set so as to strike a balance between type I and type II errors [Syrjärinne and Nummenmaa, 2016].

On the one hand, the application may raise an alarm for just about every potential deviation from the normal travel times: actual incidents will indeed be reported, together with many “false positives” where the traffic is in fact unaffected. Conversely, the application may produce very few alarms and trigger no false positives, but instead cause false negatives by ignoring some accidents and reporting only the most severe ones. The benefit of a “sensitive” system in the former case is that it reacts promptly; the “cautious” system in the latter case can be preferred for its accuracy.

Given a set of historical observations and the details of actual road accidents occurring throughout the city at this time, it is possible to observe the behaviour of the link model and calculate the total number of false and valid alarms for different parameters. With these details, it is then quite feasible to establish minimum tolerances for various characteristics, such as accuracy or response time, and select the parameters that best satisfy the established criteria.

2.2. Change point detection

For effective identification of exceptional link travel times, it is necessary to group the available observations into different classes. The set of all observations for a single day can be considered in its entirety. However, this would be an excessive generalization. Such a set includes several internally homogeneous, but statistically distinct segments, including regular traffic peaks, traffic jams and occasional road accidents. A traffic jam, for instance, may be characterized as a spell of abnormally high travel times with a gradual increase at the start and a similar decrease at the end. Each of these segments must be identified and provided with its own description: what is an exceptionally high travel time for midday traffic may be quite acceptable during rush hours. It is thus necessary to determine the time interval for each segment.

The intervals can be likened to bins (in statistical terminology), and at least two relatively simple methods make use of the concept. The equal-width bins method divides the day into equal periods of time, which is easily implemented but completely ignorant of the properties of the data (and often grossly inaccurate in practice). Indeed, it is possible to store information for each 5-minute or 10-minute interval, but this would not account for the natural distribution of rush hours, occasional traffic jams and other significant events.

The merged-bins method applies the same initial division, but proceeds to group adjacent bins with the same statistical properties [Dey et al., 2009]. A similar approach can be observed in hierarchical clustering, where isolated points are initially treated as clusters and gradually coalesce into larger clusters depending on their proximity. The merged-bins method can be quite flexible thanks to the freedom of choice between various distance measures.

More generally, the problem can be handled with methods related to time series. After all, link travel times can be represented as simple time-value pairs, with the remark that the distance between successive pairs is irregular. The most promising method in the given context is probably the concept of change point detection.

Some work in this direction dates as far back as the first half of the 20th century, such as the concept of Shewhart’s control charts [Shewhart, 1931]. For the purposes of quality control, it was originally necessary to trace a particular statistic and accurately detect the occasions of its deviation from acceptable boundaries.

In the form described by Taylor [2000], an algorithm for change point detection has been used in the implementation of the present tool. The idea behind this approach is the use of cumulative sums: given a vector of values, each element is initially replaced with the sum of itself and all preceding elements. This process will be discussed in greater detail in the following section.

Generally, change point detection is still an issue of great interest to the research community, and many different procedures have been proposed to resolve it. Takayasu

[2015] demonstrates the application of Fisher’s exact test to a time series drawn from exchange rate fluctuations, locating change points where the rate took on a principally new character. Agarwal et al. [2006] propose a simple method that takes the mean values of several points on either side of the current point, setting the change point wherever the difference of these two means reaches a local maximum. Just as Taylor, they also observe that false positives can be pruned by various tests, or policies in their terminology, such as the condition that the difference must exceed a certain fraction of the means themselves.

An extensive treatment of change point detection is found in the work of Basseville and Nikiforov [1993]. In addition to the methods already mentioned, they discuss finite moving averages, filtered derivative algorithms also encountered in image edge detection, and a number of Bayesian algorithms.

Additional results also exist on a more general level. For example, Takeuchi and Yamanishi [2006] have been able to wrap outlier detection and change point detection in a single framework. Upon replacing a time series with its respective moving averages, change points can be regarded as outliers in the resulting series. Sunehag et al. [2012] extend these findings further by clarifying that the “score” of each element will necessarily have a small expected value if the underlying distribution remains largely the same. Khan et al. [2016] treat the issue as an optimization problem, applying genetic algorithms for parameter tuning to locate multivariate change points.

Many algorithms dealing with change point detection tend to concentrate on the mean values of the original sample. This is clearly a convenient statistic to calculate, and a change in the distribution of a variable is intuitively likely to have an impact on its mean. Thankfully, all problems of this kind are in fact reducible to the case of detecting a change in mean values [Darkhovski, 1994].

No matter which procedure is used to generate a set of change points, further testing can still be performed to confirm their validity. A common tool used for this purpose is the Mann-Whitney test [Mann and Whitney, 1947]. A simple rank-based algorithm, this test effectively determines whether two data samples possess the same statistical distribution. By applying the test to every pair of intervals separated by a “candidate” change point, and merging those intervals that exhibit no significant difference, the procedure can act as a filter for change points. Conveniently, the test can be used as a benchmark for change point detection algorithms, since it can determine how many of the potential change points were in fact correctly detected. It is also a fitting choice because it does not rely on the samples being distributed normally, and can handle various sample sizes.

Once the change points have been identified, metrics can be computed for each of the resulting intervals separately. There could be, for instance, separate profiles for the morning rush hour, the afternoon traffic peak, the “calm” hours between these and the brief accidents interrupting normal traffic flow.

It is worth noting that the detection of change points is not the immediate objective of the link model. Ultimately, it must provide an adequate description of the traffic flow in a particular area, taking into account the variance of its characteristics over time. This requires accurate identification of the periods where these characteristics are clearly not the same. Simple binning methods tend to be inaccurate in such a task, and algorithms dealing with change points merely turn out to be more precise. However, their output is still not the final processing step: a model shall maintain some uniformity and generality to provide any benefit, so it can only be expected to store a restricted number of traffic profiles. These profiles should be applicable to the entire lifetime of the model, even if they do not agree with the exact change points observed at a particular time.

3. The "traffic awareness" tool

In order to implement the concept of bus links in practice, an application has been developed in order to track the links maintained in the city of Tampere, store historical data together with a reference model, and visualize the current link states at any given time.

Due to certain design constraints, the application consists of three linked components written in Java, PHP and JavaScript. In the design documentation for the system, the operation of the tool is discussed with respect to each of these components. However, in the context of this work it is more appropriate to view the same issue from a functional perspective. That is, the tool's primary functions will be covered without making detailed mappings between them and the responsible components. The application serves three distinct purposes: it continuously records travel times, maintains a global model of all observations and distributes up-to-date traffic conditions to potential clients.

Several temporary data forms and representations were needed to accomplish these tasks. Figure 2 represents these data sources and the components responsible for handling them; further discussion will occasionally refer to the terms used in this chart.

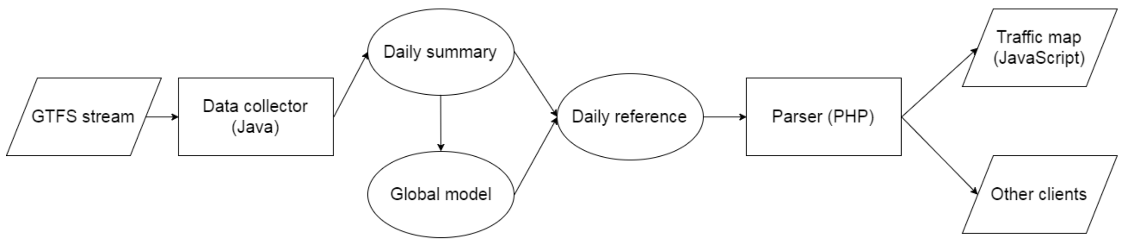

Figure 2. Data flow observed in the tool

3.1. Collecting and aggregating travel times

The backbone of the application is a data collector written in Java. Its purpose is to poll a designated service for bus messages, which are bundled and distributed in the JSON format once per second. The collector further performs certain preprocessing tasks before recording the resulting data as a sequence of comma-separated values:

3,2,0140,3A,0002,3615,TKL_93,3520,3518,1502492589012,265.0,1502492589012,265.0,46.8

Table 2 provides a brief description of these values in the same order as listed here. Some of them are taken from the original GTFS stream without modification, while some are derived from various computations. Only the components marked in bold are truly necessary for the tool's operation, but the remaining ones have proven invaluable in the debugging process.

Value	Purpose
3	Number of the bus line
2	Direction on the line, either 1 or 2
0140	Scheduled departure time from the initial bus stop (hours and minutes joined together)
3A	Full designation of the bus line. May include a letter
0002	Code of the first bus stop in the current direction
3615	Code of the last bus stop in the current direction
TKL_93	Operator of the current vehicle
3520	Code of the "next" bus stop, i.e. the one the bus is now approaching
3518	Code of the "previous" bus stop, i.e. the one most recently passed by the bus
1502492589012	Arrival time at the most recent stop (Unix timestamp with added milliseconds)
265.0	Lateness with respect to the schedule, when arriving at the most recent stop (seconds). May be negative if the bus is ahead of time
1502492589012	Departure time from the most recent stop (Unix timestamp)
265.0	Lateness with respect to the schedule, when departing from the most recent stop (seconds)
46.8	Maximum speed observed in this trip (kilometres per hour)

Table 2. Bus data gathered by the tool

Prior work in this project had already been done to implement the collector itself; the contribution of this thesis is restricted to the extension needed to handle bus links. It must also be noted that the preprocessing needed to handle the issues listed in Section 2.1.2 had also been implemented in advance, and the data used in the present work were quite regular. The collector's code thus required only a small modification. Namely, one more field was stored in every data entry, the travel time for the link at hand. The arrival and departure time may be the same, as in the example of Table 2, if a bus ignores a particular bus stop; the travel time is instead the difference between the arrival time for the current stop and the departure time for the one before it.

The collector proceeds to aggregate such entries for a period of time before writing down the complete set to a file, performing statistical processing, and returning to its original listening state. The current implementation is such that each cycle lasts a full day, beginning and ending at 2 am local time. Bus traffic at this time is either completely stopped or greatly reduced, so any processing or application maintenance can conveniently proceed with minimum data loss. While the window for such operations often lasts over an hour, only 10-12 seconds are needed in practice to handle the whole day's observations. Hence, the procedure would be equally suitable in environments with much more intense traffic, where no “silent” periods are available.

The result of this component’s operation is thus a set of links, each characterized by their stop codes and a pair of lists. The first list stores the travel times recorded for a particular link, while the second one holds the respective timestamps when those travel times were recorded, expressed in seconds elapsed from midnight for convenience. The list of travel times is then subjected to further statistical analysis.

A particular method for change point detection, which can be any procedure discussed in Section 2.2, is applied to the sample at hand. The following is a description of the originally chosen CUSUM algorithm, with the implementation adhering to Taylor’s overview [2000].

The mean value of a vector of observations is first subtracted from every element of the vector. If its cumulative sum now reaches a particularly high value before decreasing, this would likely occur due to a sequence of above-average values; a similar argument is made for the sum that reaches a minimum before increasing. The minimums and maximums of the sum are thus seen as potential change points, and the difference between them is recorded as the magnitude of change.

An intuitive understanding of this method may be aided by the example of Figure 3. Its two graphs represent a vector of observations and its cumulative sum. The vector is already scaled by subtracting its original mean value from every observation; this has no impact on the visual appearance of the values, apart from the scale of the y-axis, but is absolutely necessary for the cumulative sum to have meaningful extreme points and finish at zero. (For example, the sum of “raw” travel times data, with all observations positive, would simply grow monotonously.) The sample bears no relation to traffic data – it is in fact drawn from the study of beaver body temperatures in [Reynolds, 1994] – but still provides an insight into the algorithm’s operation. It can be seen that the minimum and maximum of the cumulative sum are indeed attained at the edges of distinct value groups, not matching the local extreme points of the original vector itself but often following them.

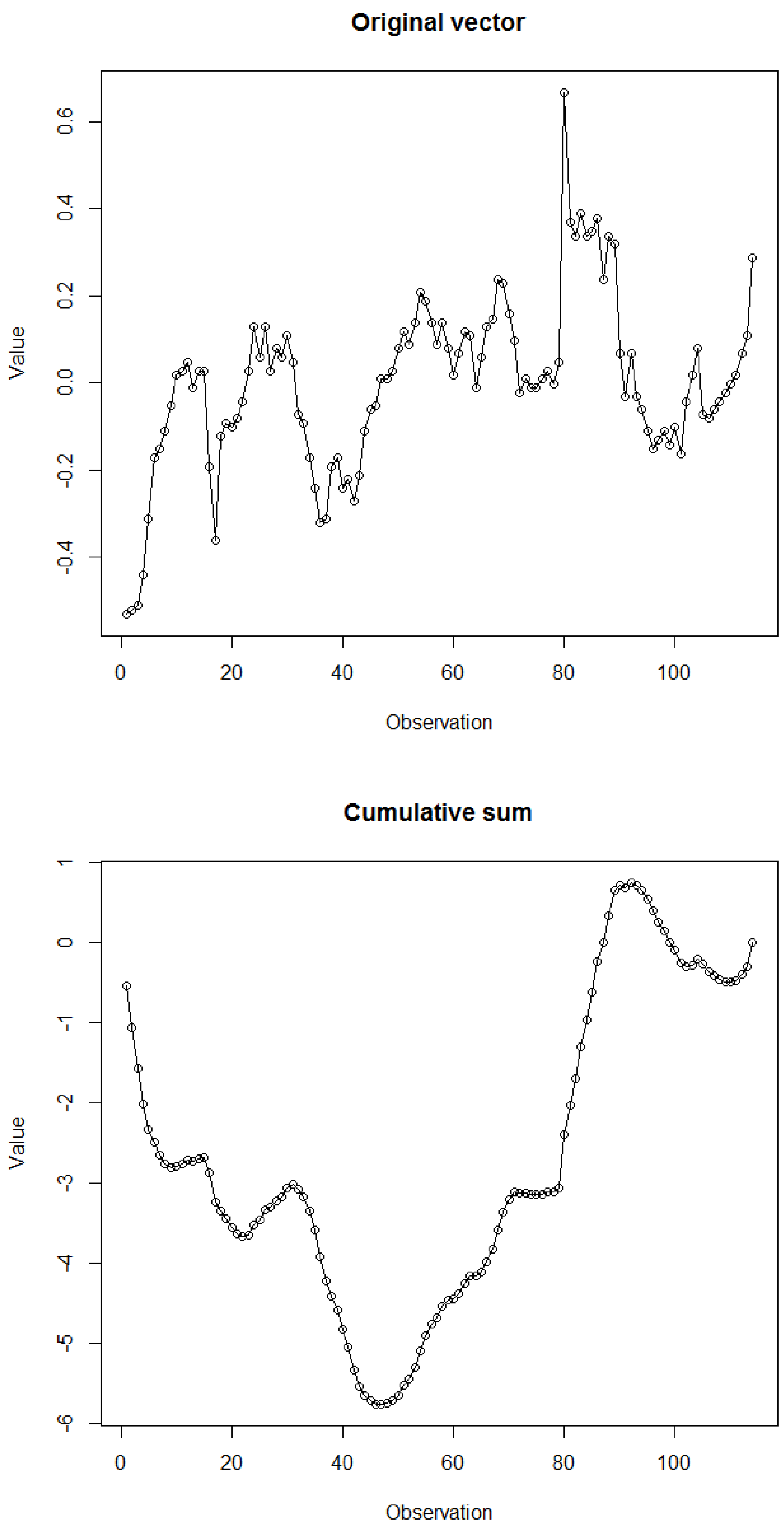

Figure 3. Initial transformations of the CUSUM algorithm

To test whether the points thus discovered actually represent any changes, bootstrap analysis is performed. The original sample, with the mean value already subtracted, is randomly shuffled a number of times, and the cumulative sum is recomputed for each shuffle. If the resulting sums generally have a smaller magnitude than that of the original sample, it can be concluded that the change point is a consistent feature of the data, not the result of a random fluctuation. The ratio of such occurrences to the total number of shuffles is regarded as a confidence level, and it should ideally reach at least 90-95%.

Once a change point is located, the same procedure can be applied recursively to the two subsamples it created on either side of itself. In terms of Figure 3, the first pass only highlights the biggest chunks of distinct values, but subsequent passes can identify smaller groups inside these chunks. The process may continue until all the subintervals produce no further change points, or until their size reaches a threshold value, such as 10 elements, at which point no meaningful change can occur. The completion of the top-level recursive iteration concludes the algorithm.

The resulting change points, no matter how they are obtained, can be stored as indices of the original list. For example, if a change point has been detected at index 22, it means that an interval is found between indices 0 and 21 (both inclusive), while the following interval starts at index 22 and extends until the following change point. For convenience, the list of change points can be immediately filled with the first and the last index of the list, even though these are formally not change points. Note that the intervals between successive observations are of random length, so every change point can be mapped to a particular observation, but not to a precise moment in time. In practice, this moment can be taken as the start or the middle of the corresponding interval.

Once the detection is complete, the initial set of travel times is broken up into several intervals. Smaller links tend to have fewer intervals, often a single one with no change points, but the exact density depends greatly on the chosen detection algorithm. In any case, separate statistics can now be calculated for each interval found. These include the median, the 90th percentile and the level – a relative measure of the link's congestion, as mentioned in Section 2.1.3. It is computed as follows:

$$l_i = 10 \ln \frac{M_i}{M}. \tag{3.1}$$

Here, M_i is the median of the corresponding interval, while M is the median of all recorded observations, which is also noted for future reference. The resulting value is rounded to the nearest integer. While this definition of the level has proven convenient, a different function may be used to express it. The level value is effectively a single characteristic of the traffic flow, which attempts to ignore the inherent randomness of individual link observations. The median is appropriate here because it is a robust estimate, one resistant to the effects that outlier values have on sample statistics. Another potentially useful function in this context is the trimmed mean.

All of these statistics, including the levels, the medians and some other information, are finally integrated into a form that might be called a “daily summary”, a JSON object containing individual link descriptions of the following kind (see Table 3 below for the descriptions of individual fields):

{"curr":31,"points":67,"prev":22,"median":56,"data":[{"u":96,"level":1,"start":349,"m":59,"end":62918},{"u":76,"level":-2,"start":62919,"m":47,"end":76126}]}

Parameter name	Purpose
curr	Code of one of the link’s endpoints
prev	Code of the link’s other endpoint
points	Number of observations recorded for the link
median	Overall median of the observations (seconds)
data	List of separate periods, one item per interval
start	Starting time of the interval (seconds from midnight)
end	Ending time of the interval, (seconds from midnight)
m	Median value of the interval (seconds)
u	90th percentile of the interval (seconds)
level	Link level for the current interval

Table 3. Contents of the “daily summary”

The “u” parameter in Table 3 is a shorthand for “upper”, the term used extensively during the development of the application. In principle, it can refer to any metric describing the largest observations of the interval, i.e. particularly high travel times. In the current implementation it is always represented by the 90th percentile of the interval in question.

The original data coming from the collector, together with the daily summaries, are saved for historical purposes. However, they are no longer necessary for the application. Instead, the values aggregated in the daily summary are used to update the global model: this is a relatively complex procedure described in the following section. Once this update is complete, the data collector discards the current observations and returns to its listening state, ready to record the new day’s travel times.

3.2. Maintaining the global model

The global model is a response to one of the major requirements posed during the development process. It stated that link characteristics were to be based on a large set of historical data. Indeed, the daily summary mentioned above already contains the medians and “upper” values for individual links, which can be seen as the criteria of traffic fluency.

However, these criteria would be more reliable, at least from a statistical viewpoint, if they were derived from observations made over a longer timeframe, such as 30 or 60 days.

At the same time, performance remained a concern throughout the development of the application. The reduction in the volume of input data had to be accompanied with an improvement in processing times. Therefore, the initial solution to the problem, the idea of scanning multiple daily summaries to extract necessary values for every link, was discarded in favour of a more efficient approach. It appeared that a single file could contain the same information in a more concise form, so that reading and updating it would be somewhat faster. This file has effectively become the global model.

Stored in binary form, the model consists of a long sequence of link tuples. Each tuple contains the two bus stop codes corresponding to the link, as well as a number of interval descriptions. Every interval contains a number of “median” and “upper” values, one per each day stored in the model. The overall number of intervals and days to be used must be set before creating the model, and has a direct impact on its size.

Mathematically, a single link tuple can be expressed as follows:

$$\langle curr, prev, m_{11}, m_{12}, \ldots, m_{1D}, u_{11}, u_{12}, \ldots, u_{1D}, m_{21}, m_{22}, \ldots, u_{ND} \rangle, \tag{3.2}$$

where

- N is the number of time periods considered in the model;
- D is the number of most recent days used when constructing the model;
- m_{ij} is the median travel time for time period i on day j ($i \in [1; N], j \in [1; D]$);
- u_{ij} is the “upper” time, the 90th percentile, for the same time period and day;
- $curr$ and $prev$ are the codes of the two bus stops defining the link in question.

Note that the boundaries of individual intervals are not specified in the model, but merely implied by other components of the application. For example, the current implementation features $N = 120$ intervals, each lasting 5 minutes, between 10 a.m. and 8 p.m. Also, the respective levels for each interval are no longer retained.

When the model is first created, every link tuple has the form $\langle curr, prev, 0, 0, \ldots, 0 \rangle$ (with a total of $2ND$ zeroes). It is thus assumed that $m_{ij} = 0$ and $u_{ij} = 0$ for all i and j. As actual travel times are processed, they begin to fill the model with real data, gradually replacing the zeroes. With such an arrangement, updating the model proves quite efficient.

The general idea is that the model stores information about D most recent days. Thus, at the end of each day the oldest values must be deleted and replaced by the newest ones in a sliding-window fashion – effectively, the tuple’s elements must be shifted one space left:

$$\begin{cases} m_{ij} = m_{i,j+1}; \\ u_{ij} = u_{i,j+1}, \end{cases} i \in [1; N], \ j \in [1; D-1]. \tag{3.3}$$

The newest values of the most recent day can now take their place as $m_{i,D}$ and $u_{i,D}$. They are collected from the structure used to generate the “daily summary”. Since the observations for every link are broken into several intervals by change points, the application shall take each of the global model's time periods, look up the interval that covers this period, and fetch that interval's median and "upper" values. Since the intervals for each link are typically much longer than the model's time periods (often an hour or more compared to the model's 5-minute step), multiple values of $m_{i,D}$ and $u_{i,D}$ end up being equal for many successive i. This is understandable, since the characteristics of a link should indeed remain the same for at least several short intervals. However, they will most likely be different for the next day.

The updates applied to every link in the global model are illustrated in Figure 4. The process is shown for median values only, but "upper" values are refreshed in the same manner. Rectangles with a solid border correspond to the initial state of the link, while those with a dashed border represent the updated state.

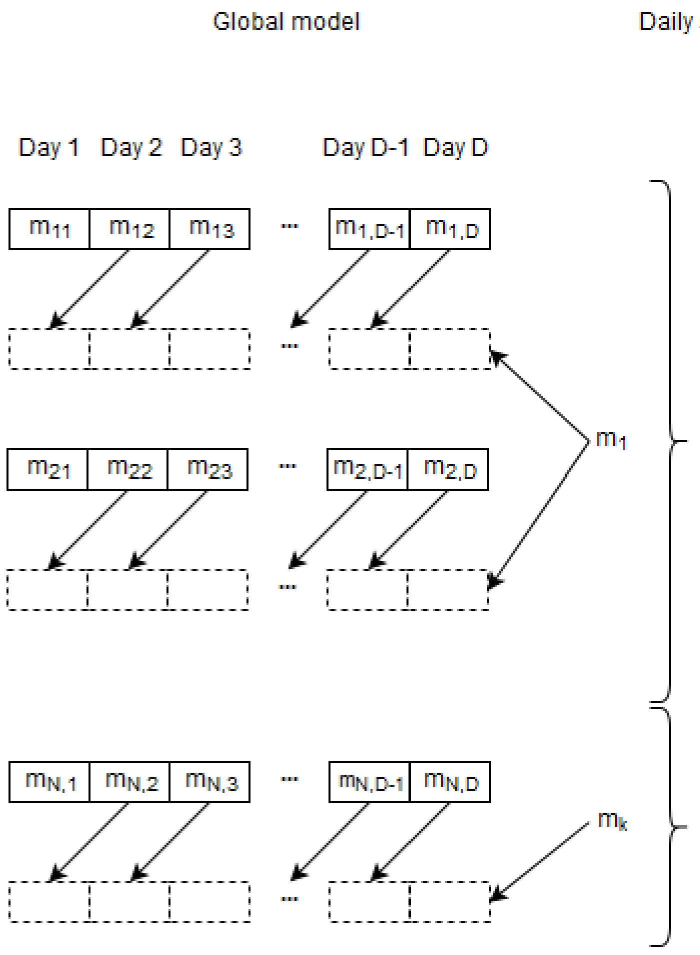

Figure 4. Updates in the global model

This update pattern allows for continuous reads and writes of relatively large fragments in the model file, so the disk operations do not produce a significant overhead. The application reads a single link's intervals, makes the adjustment mentioned in (3.3) and overwrites the original data with the result. While doing so, it also retains certain link-specific data; the purpose of this shall be explained in a while.

In addition to the performance benefit, disk space consumption is also a minor advantage of the global model approach. Unlike daily summaries, which have to be created all the time, the global model represents a file of constant size. This size can be precisely computed even before the model itself is created, and there is never a need to expand the model for any internal cause.

The most significant drawback of this approach is undoubtedly its lack of flexibility. The structure of the model depends on a known number of links, as well as the parameters D and N. If these parameters have to be changed, or some links added or removed, the entire model must be reconstructed from scratch. Even in ideal conditions, this has to take place whenever the timetables are changed, since new schedules most likely involve the use of different links, those not present in the current model. For the city of Tampere, such a change occurs twice every year.

In principle, introducing some additional complexity into the model’s layout could mitigate all of these problems. Unused links can be stored in the same model file, and a separate attribute may be added to mark whether a particular link is currently in use. The application will then be able to skip whole portions of the model file, and restoring a previously “disabled” link will be as simple as inverting its mark.

Most other changes to the physical link layout, in practice, are instances of either merging two links, when their common bus stop is removed, or splitting an existing link, when a new bus stop is added within it. In the former case, the combined link’s median and “upper” values are obtained by simply summing the respective values of the two original links. The latter case is not as straightforward, but a reasonable approximation may be derived by interpolating the original travel times, taking the relative length of the newly split parts into account. This approximation will be further refined as new observations accumulate in the model. Redundant links from previous merge operations can be reused to store newly split ones, to avoid expanding the model, or simply purged from the file at regular intervals.

In any case, the generation of a new model is also a fairly simple task. After all, the initial state of such a model is merely a zero-filled template of the appropriate size. Once it is produced, the model will slowly fill up with data as new daily summaries are formed. If these already exist, and the model is being created some time after the start of a new period, the summaries can be immediately used to saturate the model.

The greatest concern in the generation of a model, assuming the inflexible approach originally taken in its creation, is perhaps the choice of links to be included in it. After all, these links must be representative of the traffic situation, since no links can be added or removed once the model is in use. Adding every possible link to the model would be wasteful: most of the file would be occupied by "minor" links that are hardly ever traversed. Adding only the links used in a single day would be insufficient, since some important links only utilized on specific days would thus be excluded. A compromise solution adopted for the implementation involved a seven-day set of links. Effectively, seven files containing travel times for a particular week were merged together, a link structure was built on the basis of that file, and every link featuring at least 10 observations was included in the model. This produced a global model file of roughly 50 MB in size, and processing it takes only a few seconds.

This is an overview of how the global model integrates link parameters over many days, but not of exactly how these parameters determine the standards for link fluency (in a way, future values based on past values). The process is in fact quite straightforward: every link is given a single median and "upper" value for every interval stored in the model. The median is taken as the median value of all the D link medians (those applicable to the same interval, but on different days), while the "upper" value is similarly the median of the D "upper" values. This is precisely the processing done by the application before writing the updated model data.

An example can be easily provided with a single link tuple. It has the following form, expanded here into a matrix for clarity:

$$\begin{pmatrix} m_{1,1} & m_{1,2} & \cdots & m_{1,D-1} & m_{1,D} \\ u_{1,1} & u_{1,2} & \cdots & u_{1,D-1} & u_{1,D} \\ m_{2,1} & m_{2,2} & \cdots & m_{2,D-1} & m_{2,D} \\ \cdots & \cdots & \cdots & \cdots & \cdots \\ u_{N,1} & u_{N,2} & \cdots & u_{N,D-1} & u_{N,D} \end{pmatrix}. \tag{3.4}$$

We assume here that the current day's values have already been added into the rightmost column of the matrix, superseding the oldest values of the model. For convenience, the application also makes a copy of each row of the matrix and sorts it in ascending order. The link's median and "upper" value for a particular interval i can then be calculated thus:

$$\begin{aligned} \mu_i &= m_{i,D/2}; \\ \upsilon_i &= u_{i,D/2}. \end{aligned} \tag{3.5}$$

However, if the model has been in use for fewer than D days, or if some links have not been navigated at least once in a particular interval, the model will contain zeroes instead of actual travel times. These should not affect the calculation. To avoid their influence, the following adjustment is made:

$$\begin{aligned} \mu_i &= m_{i,(C+D)/2}; \\ \upsilon_i &= u_{i,(C+D)/2}, \end{aligned} \tag{3.6}$$

where C is the number of zeroes encountered in a given interval.

This notation readily permits the usage of other methods of arriving at the final values. For example, the following expression replaces the median with the mean:

$$\mu_i = \frac{\sum_{m_{ij} \neq 0} m_{ij}}{D - C}. \tag{3.7}$$

It is equally easy to use a weighting scheme in the calculation, as suggested in Section 2.1.3. In particular, exponential weighting can be implemented thus:

$$\mu_i = (1-\lambda)\sum_{j=1}^{D} m_{ij}\lambda^{D-j}, \ 0 < \lambda < 1. \tag{3.8}$$

The values should no longer be sorted in this last case, so that the most recent ones receive the greatest weights, and sorting is also needless when (3.7) is used. The initial $(1-\lambda)$ term is the inverse sum of all weights, or at least a close approximation of it for $D \geq 50$.

In any case, the global model is utilized to produce a sequence of N pairs of median and "upper" values for each link, with every pair corresponding to a single interval. These values are wrapped in a JSON object and stored as the reference for link fluency, together with the overall median travel time (not included in the model itself). Given the span of the global model, the values can be seen as averages over a long period of time. The reference is valid for one day, until the next update of the global model, when it is recalculated.

In principle, the global model alone determines the contents of each particular daily reference. However, Figure 2 suggests that the daily summary also makes its own contribution. This is because the summary yields the most recent median and "upper" travel times for every link, which are added to the global model while the oldest ones are removed from it. Once the link information is thus refreshed, it is practically simpler to use the global model as the only source for the daily reference.

3.3. Determining current traffic conditions

No matter how the reference values were obtained, the users of the tool are interested in the actual state of each link derived from those values. A PHP parser, functioning as a separate component of the application, is used for this purpose. As the script is queried, it looks up every link in order, finds the interval corresponding to the time of the query, and retrieves the respective pair of μ and υ. These values represent the typical and the "upper" travel time for the link at the precise moment when the user's request is made. In the case of the model presently implemented, as described in the previous section, if the script were launched at 3:03 p.m., it would look up the values corresponding to the interval between 3:00 and 3:05, i.e. between 54000 and 54300 seconds since midnight. Since the first interval starts at 10 am, the values returned for each link would correspond to μ_{61} and υ_{61}.

In addition to these, the script accesses the same file that is used by the data collector to fetch the actual travel times. This turns out to be a somewhat time-consuming procedure. For every link, the most recent travel time must be chosen; if the dataset includes multiple observations, as is usually the case, only the latest one is of interest. At the same time, messages from different buses are processed and recorded in the order of their arrival, so there is no internal pattern in their sequence. No matter which fragment of the dataset is scanned, it is almost impossible to tell whether the observation for a particular link is the most recent one, and if not, how soon the next one will be encountered.

Therefore, the approach taken in the present implementation is simply to scan some distance back from the end of the file and process all the observations in the resulting segment. They have a natural chronological ordering, so every new observation for a link already encountered can be safely used to overwrite the older one. How far to rewind the file is a tricky question: the script must capture at least one observation, even for links not commonly traversed, but it cannot waste too much time going through data that will be immediately replaced by more recent values. In practice, a segment of 800000 characters is chosen for scanning, which corresponds to about an hour of traffic – and thus to at least one or two observations on even the least frequent bus lines.

At this point, the script has a complete description of the current traffic state. Each link originally present in the global model is characterized with four values:

- the median travel time for the current interval, m;
- the "upper" travel time for the current interval, u;
- the median travel time for the entire link, med;
- the actual travel time most recently observed, t.

Initially, the script was supposed to determine the link's state based on these four parameters and include that in its output. However, potential clients should not necessarily accept one and the same decision; instead, they should be given the right to use their own criteria. As a result, the script merely delivers the values of interest, while each client is free to interpret them independently. This separation of concerns is highlighted by Figure 5.

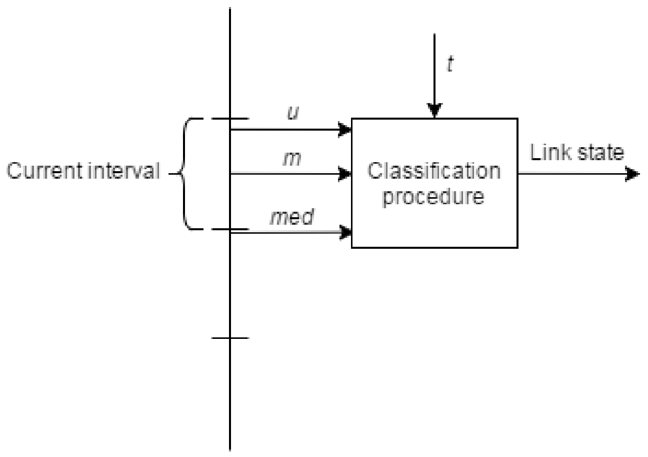

Figure 5. Derivation of link states

A visualization has been created as an example client for the application, and it features its own decision-making process. The visualization in question has been implemented as a map of the city of Tampere, using the facilities provided by the Google Maps API. Prior work on this project has already produced an extensive set of coordinates for most of the individual links. Each link is described by a collection of latitude-longitude pairs, usually between 5 and 15 per link; drawing lines between pairs of adjacent points thus defined yields a sufficiently accurate depiction of the link.

Unfortunately, any changes in the position of bus stops will cause the resulting image to diverge from the true boundaries of the respective links. This is not a significant issue, since the bus stops only correspond to the first and the last point in the list of coordinates. Intermediate points will not change at all, as long as the bus line itself is not directed along a different route. When introducing new links, such points should be placed on the intersections where the vehicles change their direction, so that straight-line approximations will remain accurate enough. This also reduces the number of coordinate pairs that need to be looked up.

Upon receiving a JSON package from the PHP script described in this section, the browser proceeds to display the links on top of the map. Figure 6 represents a possible visualization generated by this process. The image always corresponds to the latest data available when the page was loaded, but does not update in real time: this would require intensive communication with the server, with the browser constantly receiving many small updates about various links that would most likely not change their states.

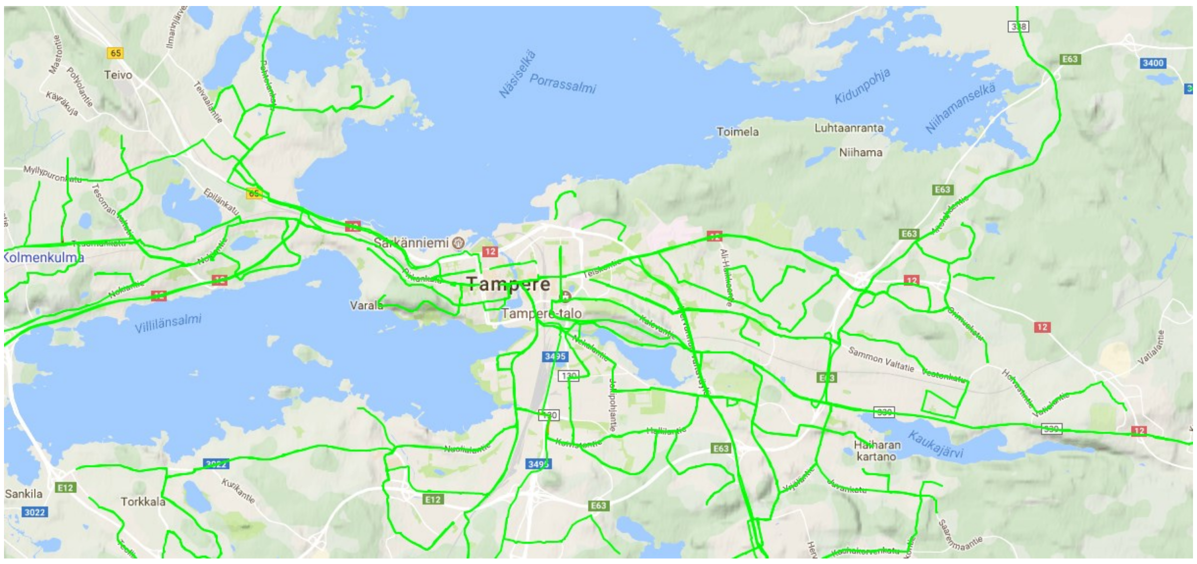

Figure 6. A sample visualization

The colour of the lines used to depict each link depends on that link's state, which is determined as follows:

- If $t > k \cdot u$, the state is "exception". Such links are drawn in red.

- If $t > M \cdot med$, the state is “regular congestion”. Such links are shown in yellow.
- Otherwise the state is “fluent”. These links are the most common, and the colour used for them is green.

Here, k is a constant set to 1.5. Likewise M is a constant set to 2. Note that, since u depends on the time of the day while med does not, the congestion threshold is fixed, while the exceptional threshold is time-dependent. A link could theoretically be in an exceptional state without meeting the condition for “regular congestion”, but this is extremely unlikely since u is much higher than med.

This classification is entirely based on the recommendations given in [Syrjärinne and Nummenmaa, 2016]. To reiterate, other clients are free to introduce their own decision logic, which may be more or less detailed than the one shown here. They may find it appropriate to deal with more or fewer distinct states, or to alter the criteria for deciding between them. For example, it may be fitting to make use of the m parameter, which is not employed in the current classification.

4. Experiments

The performance and functionality of the tool described in Section 3 have adequately met the initially proposed requirements. However, a promising direction for future research was seen in the implementation of a particular procedure, driven rather by practical considerations. This procedure handled the issue of change point detection in travel times, i.e. the detection of those moments when the traffic flow took on a different nature. Existing sources have suggested a diverse group of algorithms designed to tackle this problem. It was naturally attractive to implement a number of them and apply these within the available tool.

4.1. Purpose

It would certainly be tempting to choose the "best" possible method, defining a number of quality criteria and selecting the one that matches them better than any other. However, producing verifiable criteria is rather difficult in this context, apart perhaps from the obvious standard of performance. Indeed, the relative execution speed on the same machine, using the same benchmark dataset, is easily measurable. Yet the introduction of other aspects soon leads to more awkward considerations.

For example, it is appropriate to evaluate the accuracy of a change point method. It may be defined as the ratio between the number of initially detected change points and the number of actual change points remaining after filtering, averaged across all the links within the dataset. As was mentioned above, the Mann-Whitney test will be applied as a filtering procedure that may discard some of the candidate points proposed by the detection method. After all, the problem of change point detection does not have a single correct answer: different algorithms may yield radically different sets of change points. This is acceptable, as long as the test does not suggest that the intervals between these change points actually have a similar nature.

This scheme will generally penalize methods producing too many incorrect change points (incorrect in the sense that they do not separate statistically distinct subsamples). This is also slightly flawed in a way: a "conservative" algorithm that generates only a few points, making one or two mistakes, will likely score better than an algorithm finding all the correct points together with multiple "false positives". Such a tendency may be acceptable. But how would this evaluation handle the dependencies of change points on each other? If a filtered change point causes several points after it to fail the test as well, while without filtering they would be considered "true", would the procedure not penalize the method repeatedly for the same mistake?

Various workarounds can still be utilized here. Firstly, the verification process can ignore all detected change points between a false positive and the next confirmed change point, considering them to be caused by the false positive in question. This may result in discarding almost all of the candidate points, thus producing an unreliable accuracy measure. Secondly, the procedure can temporarily assume that a false positive is in fact a true change point (while still counting it as an error) and check further change points with respect to this assumption. This is perhaps a more reliable option, but it features a slightly more complex implementation, especially if attempts are made to isolate the effects of multiple false positives in a row. In any case, it is difficult to arrive at a straightforward, concise definition of the accuracy metric.

To avoid these concerns, further experiments have approached the problem with a purely pragmatic intent. The objective is simply to discover which change point detection algorithms are the most "convenient" in the setting of the current tool, without judging them as generally better or worse. Since the primary context of the application is visualization, the resulting distribution will be mostly viewed from this angle.

The most significant characteristic observed here is likely the density of detected change points. If the intervals delimited by them only last a few minutes, the resulting image will be hard to interpret: travel times will be frequently evaluated with respect to varying medians and "upper" values of each new timeframe, causing link states to change frequently, while the actual traffic flow may in fact be virtually the same. If the change points are too sparse, perhaps one per day, the link states will hardly ever change, and the very purpose of the complex link model will be brought into question. This may be understandable for remote links, where bus traffic is indeed nearly constant throughout the day, but somewhat unnatural for the congested links in the city centre. Admittedly, a different environment could well require a different distribution of change points.

Finally, the detection methods considered in this analysis are typically configurable to be more or less sensitive to change points. This is often not a direct goal of their design, but a natural effect produced by their respective parameters. For the comparisons of this section, the methods shall be adjusted to produce change points with various density. It is preliminarily expected that a single day's data over a busy link should contain roughly 10 such points: there are several natural traffic peaks over this period, and perhaps a few more artificial irregularities, but the character of the traffic flow is unlikely to change more frequently than once per hour.

4.2. Data

Ultimately, the dataset chosen for experimental purposes consisted of a single day's observations. On the one hand, this is a fairly large sample, consisting of about ninety thousand travel times distributed across a thousand links. Any irregularities, whether due to measuring errors or to outlier values found within certain links, are most likely compensated by the sheer number of calculations required. On the other hand, expanding the dataset further would not introduce any valuable information into it: the sample already contained the vast majority of available links, excluding a few remote ones only used occasionally during weekends.

Extending the sample to several consecutive days would also be unproductive. A single day is certainly a cyclic period in terms of traffic flow, and apart from occasional accidents, each new day exhibits largely the same rush periods and quiet times. Moreover, bus traffic at the end of each recording cycle is as subdued as at the beginning of the new cycle. Thus, concatenating the observations from multiple days would simply create more extended "link histories" without introducing principally new change point configurations.

4.3. Methods

As described in Section 3.1, the original mechanism for change point detection followed the procedure laid out by Taylor [2000]. This algorithm relies on replacing a sequence of observations with their cumulative sums (for brevity, it will be further referred to as the CUSUM method). The minimum and maximum of these sums is produced by a string of particularly low or high values, respectively, and their difference is viewed as a significant trait of the original sample. If the sample is randomly shuffled a large number of times, and the resulting difference is smaller in a significant share of cases p, then the initial difference is considered to be a product of true change, as opposed to random variations, and thus the maximum of the cumulative sum (more precisely, its position) is declared a candidate change point. Here, p acts as the confidence of this decision. The cumulative sums and permutations are then repeated recursively on either side of the candidate points, as long as this process keeps yielding new candidates.

The procedure is thus governed by the number of shuffles N, as well as the fraction p of those shuffles where a smaller difference must appear to obtain a change point. N has a notable impact on the performance of the algorithm, since the cost of finding a single change point for a link with m observations, given the need of computing cumulative sums and normalizing them, is effectively $O(mN)$. The current implementation uses a fixed number of 500 shuffles. With hundreds of links and a minimum of 10-20 observations per link, the search for other methods of change point detection was at least partly motivated by this performance concern. At the same time, p is expected to influence the accuracy of the method, since it directly governs the confidence with which a change point may be proposed. In many statistical tests of significance, a value of 0.95 or even 0.99 is appropriate; in the current setting, $p = 0.8$ is needed to produce a reasonable number of change points.

A much lighter algorithm fit for consideration is drawn from Agarwal et al. [2006]. This particular method is additionally suitable for streaming contexts, since it effectively operates on a window featuring only a limited number of observations, and will thus be cited as the sliding-window method. For a given window size N, it chooses a central point and computes the mean values for the points to its left and to its right (or, more precisely, the difference of these means). Once every point has assumed the role of the central point, the observation with the highest difference of means is regarded as the candidate change point. This is sensible, because the change point is precisely where the observations change their nature, and are thus the most likely to produce different means afterwards. Considerations of monotony allow the mean to be replaced with just the sum of the entries, and even the sum itself is recomputed just by looking at the values on either end of the window.

The method is thus guided by the value of N. The paper in question demonstrates that increasing N reduces the probability of false positives discovered. However, given that the procedure effectively chooses a local maximum, it shall produce a candidate change point for every $2N$ observations in the sample. Such regularity is most likely unnatural. Therefore, a second parameter p is proposed by the authors as an additional constraint: a change point is only observed when the difference between the means exceeds the "left mean", multiplied by p. This effectively reduces the algorithm's sensitivity to small changes. A value of 0.3 to 0.4 is recommended in the article.

Additional comparisons will be made with the equal-width bins algorithm, which simply sets change points at a fixed distance in time from each other without considering the actual observations in any way. Since a filtering procedure is run afterwards, likely eliminating most of these points, testing the method will provide a demonstration of that procedure's effectiveness. The algorithm allows the user to configure the size of each bin, and thus the resulting change point density, but this parameter is not expected to have a significant impact.

The merged-bins algorithm, a reasonable extension of the preceding method, is not directly implemented for this analysis. However, in practice it is implicitly performed by the filtering procedure for every other algorithm. The Mann-Whitney test removes those change points that actually separate statistically similar intervals, joining them together; in the same way, the merged-bins method glues adjacent intervals together based on a similarity criterion. Notably, it will never "correct" a change point or introduce one in a new position, but merely retain or discard already existing ones.

4.4. Findings

Since the number of links in the sample is relatively high (1404), it is not feasible to observe them individually. Instead, the discussion will focus on characteristics exhibited by the entire group. Figure 7 describes the dataset in terms of the observations recorded for various links, as well as the timeframe covered by these observations.

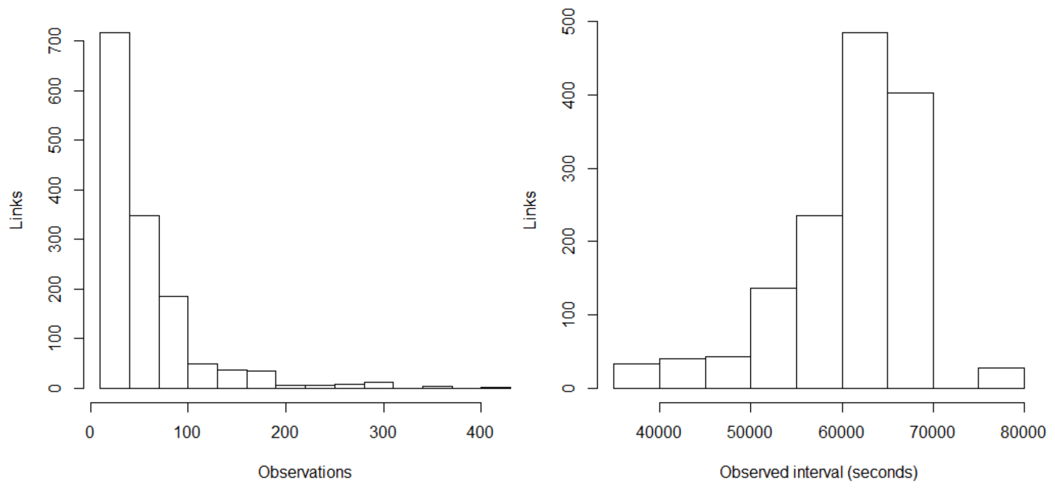

Figure 7. Characteristics of the sample

It appears that the sample is dominated by links with at most 50-60 observations, which provide less material for change point detection. These most likely correspond to bus stops belonging to infrequent routes, where bus traffic is relatively sparse. The timespan between the first and the last observation, corresponding to the right half of the figure, instead depends on the timetable observed by the respective routes. While uncommon bus lines are often operated for a shorter time, and sometimes left completely unused on certain days, the connection between the intensity of a line and the duration of its observation period is not as immediate.

Starting with the CUSUM method, it is instructive to observe the influence of its parameter, p, on the number of change points produced, as shown by Figure 8. The raw output is examined here without any further filtering.

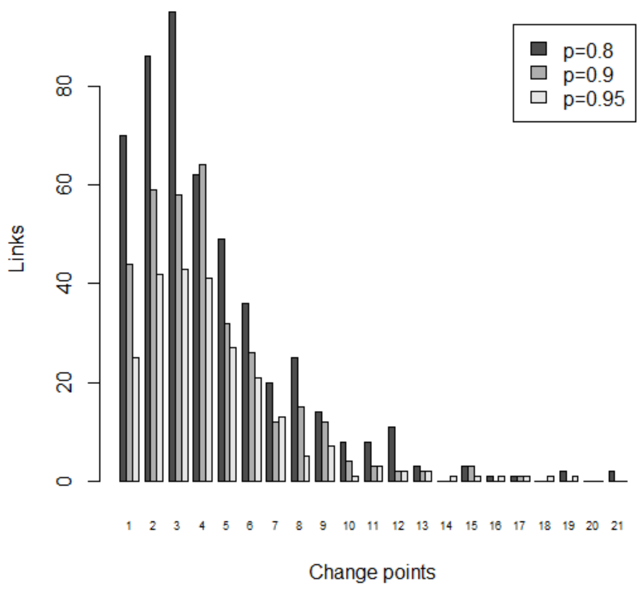

Figure 8. Change point distribution based on p

The case of 0 change points was deliberately omitted from the graph for scaling reasons: links with no change points totaled 907, 1067 and 1166 respectively. In every case, the obtained distribution closely matches the original distribution of the observation counts: it is reasonable to assume that the number of change points is proportional to the number of initially recorded observations, and the two variables indeed display a correlation of about 0.55.

As expected, the number of change points diminishes as the confidence level set for the CUSUM algorithm rises. There are occasional exceptions for links with many change points, where an unexpectedly high value can be produced even with a high confidence level, but this is explained by the randomness inherent in the method. It can also be assumed that the reduced number of change points will be less affected by further filtering procedures. However, it is also clear that the value of p must be restricted to at most 0.9: in the third case of Figure 8, the share of links with no change points at all rises above 80%. Since the filtering process cannot introduce new ones, the resulting distribution is too poor to describe the traffic situation adequately.

The exact effects of filtering are shown in Figure 9 below. The CUSUM algorithm was run with the same three confidence levels as before, and the outputs derived from it were then passed through the parameterless Mann-Whitney test.

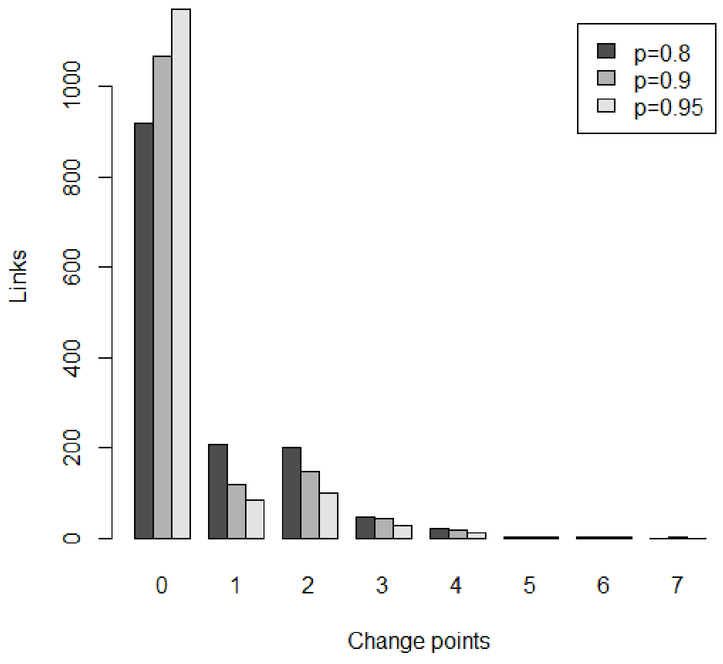

Figure 9. Change point distribution based on p (after filtering)

The process displays a moderating effect in all three cases: the maximum number of change points found in any link is down to 7, and values above 4 are already extremely unlikely. As before, an increase of p corresponds to a decrease in the change point counts;

the column for 0 now aggregates links that never had any change points and those that lost them after filtering.

The equal-width bins method serves as a useful background for the filtering process. Since it adds no “logic” to the process of change point detection, other than spacing them at roughly equal distances, the emphasis placed on further processing greatly increases. Figure 10 illustrates the initial change point distribution produced by the algorithm with $N = 600$, i.e. setting a new change point every 10 minutes.

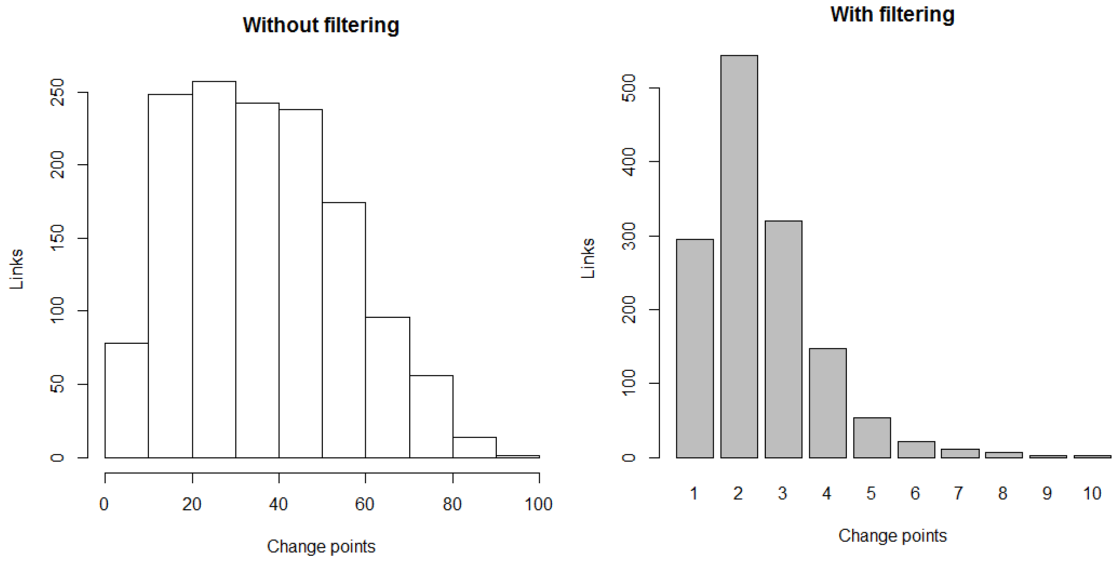

Figure 10. Effects of filtering with the equal-width bins algorithm

The layout of the left graph would be expected to resemble the one in the right half of Figure 7, since the number of change points found by the current algorithm scales directly with the length of the link’s observation period. The differences must be attributed to the fact that actual observations are placed unevenly in this period: if the intervals between them are generally shorter than N, more change points will be produced. However, a more sparse distribution is equally possible, regardless of the distance between the starting and the ending moment.

The effects seen in the graph on the right are, however, less obvious. The distribution is similar to the ones seen before, with 2 change points being the most common occurrence. However, this time the zero column is completely absent: every link has remained with at least one change point, and the share of those with at most five is very significant. On the other hand, the filtering procedure has still eliminated large sequences of change points, so that groups of 8 or more in a single link are rare.

For the sliding-window method, it was initially necessary to decide the window size denoted by N. The original article describing the algorithm suggests a value as low as 15-20, trying to reduce the time lag arising in streaming contexts. However, the limited number of available observations per link (see Figure 7) demanded a further reduction of this parameter. The central points established by the sliding window cannot fall in the first or the last N points of the link; moreover, the local maxima of the resulting sequence of "mean differences" are also taken with a gap of at least N points. With the total observation count labelled O, the algorithm can produce at most $(O-2N)/N$ change points, which means at most 3 or 4 for the majority of the original links. A similar effect can be produced by increasing the CUSUM algorithm's confidence setting.

For this reason, the method was configured to run with window sizes of 5 and 10. Another consequence of this reduction is the increased likelihood of false positives: the window size limits the "scope" of the algorithm and prevents it from detecting long streaks of increasing or decreasing values. However, it was hoped that the filtering phase would eventually discard spurious candidates. Figure 11 illustrates the resulting change point distribution.

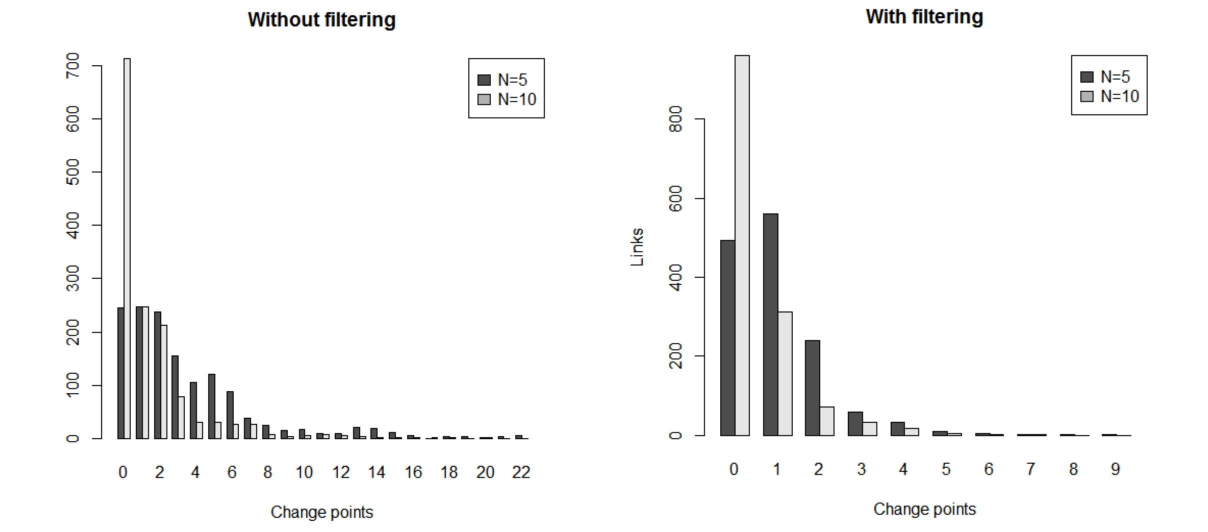

Figure 11. Effects of filtering with the sliding-window algorithm

Evidently, the value of 10 for N is close to its practical maximum in the current context, since more than half of all the links yield no change points whatsoever. Apart from this spike, the algorithm understandably finds more candidate points than the CUSUM method. Like the equal-width bins algorithm, it will always discover more candidates in links with many observations, as seen in the long tail in the left half of Figure 11. The filtering sequence also tends to leave slightly more verified change points, but links with 5 or more of them are still extremely rare.

The general scarcity of detected change points also suggests that applying further constraints to the criterion of the sliding-window method, such as requiring the difference between the "left mean" and the "right mean" to exceed a certain threshold, would not be appropriate here. Such conditions would further diminish the number of candidate points in an otherwise small sample. They are perhaps more fitting in samples with hundreds or thousands of observations, in streaming contexts, as demonstrated in the original paper, and in the absence of any other filtering techniques.

It appears that the Mann-Whitney test shows consistent results regardless of the method initially used to generate change points, reducing them to 3-5 or fewer per link in most cases. However, its inability to introduce new change points is apparently complemented by the equal-width bins algorithm. In the mass of points it produces, at least one is reliably placed so that it persists even after the filtering phase, although the vast majority of them are of course spurious. Conversely, the CUSUM algorithm can yield very few change points by itself, and the filtering process is unable to alter the situation afterwards. This is due to the algorithm's restriction of producing at most one change point in the interval it considers: once the point is set, further analysis concentrates on the subintervals it generates, and other potential candidates in the original interval are discarded.

In any case, the number of change points remaining after the entire processing sequence will rarely exceed 5 or 6. This limit is lower than originally hypothesized; it would correspond to the beginning and the end of one or two "exceptional" traffic periods, with a few gradual, one-sided changes of the traffic flow between them. The pattern of two rush hours, one in the morning and one towards the evening, may well be applicable here.

5. Conclusions

This chapter summarizes the concepts discussed up to this point. Together with a brief overview of the implementation, it recaps the answers to the research questions posed at the beginning of this thesis. In particular, the following subsection outlines a boundary between various traffic states, as defined for a single link, as well as the outcomes of change point detection and the structure of a long-term link model. Furthermore, some consideration is given to the restrictions inherent in the work and to a number of directions in which it could be further developed and extended.

5.1. Results

The concept of bus links has now acquired a practical realization. At present, it is a visualization tool represented by a Web application. The application reads expected link travel times from a model file, collects the most recent observations from the city streets and displays each link on top of a city map, coloured in accordance with its state (fluent traffic, congestion or an exceptional situation). This visualization is supported by the Google Maps API, and extending it further should not prove difficult. Data packages needed to determine link states are transmitted in the widespread JSON format. Additional clients that require the same data for their own purposes should benefit from the ease of handling this format.

How to determine the state of a particular link was an open question relevant to this thesis. The proposed solution requires two external values: the overall median travel time and the “upper” travel time, or the 90th percentile of all times in a given interval. If the most recent travel time over the link exceeds the median, multiplied by a specific value, the link is regarded as congested. If the travel time exceeds the “upper” value, again increased by a multiplier, the link is considered to be in an exceptional state. When neither of these conditions holds, the link is viewed as fluent. The recommended magnitudes of both multipliers, as well as the effects of their variance, have been discussed in earlier literature, and this work follows the suggestions provided there.

Since the idea of bus links was to cut down processing times by discarding much of the input data, performance was a key issue in the development process. One of the bottlenecks identified during development was the necessity to maintain a long-term model that would retain average and “upper” link travel times for an extended period, such as 30 or 60 days. These times would then be averaged to provide more accurate estimates for the present-day model. However, scanning multiple files to gather the values required was considered too time-consuming. Instead, a single file was introduced to hold all the values in a tightly packed sequence. Reading and updating them proved very efficient: the file represents a sliding-window system that automatically overwrites the oldest values with newly derived ones, retaining all the travel times over a predefined timeframe.

The statistical problem of change point detection also played an important part in the work. Within the tool, it was used to examine and break down sequences of daily travel times for individual links. Different times of the day were expected to have a different nature of traffic flow, and thus required distinct averages. The procedure originally employed to detect the change points between these periods was based on cumulative sums and bootstrapping, a computationally intensive algorithm involving repeated random permutations of the input data.

Experiments indicate that the CUSUM algorithm is nonetheless an adequate choice for the task, as long as its confidence value is set between 0.8 and 0.9. Higher settings make the method "stricter" and greatly reduce the number of change points it produces. Given the presence of a filtering stage at the end, other methods with less internal logic are in fact equally applicable. Even the equal-width bins algorithm, which does not concern itself with any statistical properties of the data, ends up producing a fairly balanced change point distribution.

The Mann-Whitney test, acting as a filtering procedure for the candidate change points produced by other algorithms, has produced largely the same impact in all cases. By comparing the subsamples on either side of a potential change point, and removing the point when they in fact belonged to the same population, it greatly reduced the influence of randomness on the detection algorithms. It is still possible for a link to have several, equivalently valid change point distributions, but at least their overall quantity has been normalized by the filtering process.

5.2. Restrictions

The visualization in question always provides the link state as determined by its last recorded observation, which is not necessarily useful. If the link's condition was regarded as fluent four hours ago, and no bus has since travelled across that link, the data point may well be considered obsolete. Thus, there should be some way of differentiating between recent observations and more dated ones. One approach is to use lighter and darker shades of the three main colours used on the map (assuming the default three-state classification used at present), but these may be hard to tell apart. A better way is perhaps to utilize different line styles, such as dotted or dashed lines, to mark the links. The support for such graphics is provided by the API, yet no practical experiments have been performed in this area.

The data formats used in the project were largely constrained by the structure of public transport data utilized by the application and the preprocessing already done by existing code. The fields in the resulting output are only those of interest to various users and those available to the tool itself. Given a different source of input data, which was not assumed for this work, these very fields might not be possible to compute any longer. It remains to be determined how these outputs may be replaced and under what circumstances.

Although the tool itself is quite simple to operate, deploying it in an environment with a more restricted infrastructure may be challenging. The coordinates of the area's bus stops must be carefully mapped and their possible pairs established for the models to work smoothly. Moreover, it may take a long time to observe the climatic conditions of the environment and determine the optimal periods for refreshing the link model, especially with the additional constraints imposed by the public transport itself.

5.3. Future work

This practical implementation of the bus link concept is presently restricted to just one client and a single requirement. While a visualization tool may seem the most attractive of clients, many other applications could be conceived that would need the same data for further analysis without displaying them immediately. For example, an application could accumulate the link states at fixed intervals over several days, perhaps even weeks, and then display the evolution of those states in its own fashion. Such clients will conveniently be able to make use of the same data packages. However, it is also possible that their development will discover flaws in the current data format that were not revealed in the context of the present needs.

Apart from the question of performance, the behaviour of change point detection algorithms when applied to larger samples also represents a fruitful direction for further investigations. The filtering procedure used in the current study consistently reduced the number of change points to 1-4 per link, which is likely an accurate reflection of the typical traffic flow. Moreover, an inappropriate configuration of certain algorithms can produce a poor change point distribution even before the filtering process, leaving most links without a single change point. Both these effects are at least partly caused by the limited number of initial observations serving as the input, and increasing this quantity to 300-500 may well produce a different behaviour.

The rigid structure of the global model has been adopted partly because the changes in the bus link structure are rare in the present environment. However, this is not always the case. While some initial approaches to a more flexible solution were laid out when discussing the model, they have not yet been put to practical use. More volatile environments, where bus stops are frequently moved or taken out of use, would prove useful in testing these suggestions.

At present, human interaction is currently required to make any changes, even the most trivial ones, to the existing models. A logical next step would be the automation of certain changes, possibly by converting them into commands accepted by the tool. The software would be responsible for maintaining the integrity of the link models upon the processing of every individual command, reducing the possibility of error, much like a database management system. An authorization mechanism may also be needed to ensure that such commands can only be issued by the tool’s maintainers. However, their task can still be simplified: for example, the official channel notifying the public of changes in the bus stop network can be scanned for new messages, which would then be converted into update commands.

Another potential direction for further development is the concern for scalability. It is unlikely that the bus network of Tampere will expand so much as to decrease the performance of the application below acceptable levels; indeed, the solutions proposed in this thesis ensure that the input data are greatly reduced in volume, and their flow has little impact on the tool’s running time. However, adapting the same product in areas with greater population density and more extensive public transport infrastructures might raise some issues. After all, some cities have 50 to 100 times more buses navigating the streets simultaneously. A remedy to these issues can be found in the big data solutions of today, such as the freely available Hadoop implementation. Designed specifically to ensure scalability with large data volumes, it is likely to provide rapid processing no matter the environment involved.

References

[Agarwal et al., 2006] Manoj K Agarwal, Manish Gupta, Vijay Mann, Narendran Sachindran, Nikos Anerousis, and Lily Mummert. 2006. Problem determination in enterprise middleware systems using change point correlation of time series data. In: *2006 IEEE/IFIP Network Operations and Management Symposium (NOMS)*, 471-482.

[Baptista et al., 2012] Arthur T. Baptista, Eric P. Bouillet, and Pascal Pompey. 2012. Towards an uncertainty aware short-term travel time prediction using GPS bus data: case study in Dublin. In: *2012 15th International IEEE Conference on Intelligent Transportation Systems*, 1620-1625.

[Barabino et al., 2013] Benedetto Barabino, Massimo Di Francesco, and Sara Mozzoni. 2013. Regularity analysis on bus networks and route directions by automatic vehicle location raw data. In: *IET Intelligent Transport Systems*, 7 (4), 473-480.

[Basseville and Nikiforov, 1993] Michèle Basseville and Igor Nikiforov. *Detection of Abrupt Changes: Theory and Application*. Prentice Hall, 1993.

[Darkhovski, 1994] Boris S. Darkhovski. Nonparametric methods in change-point problems: a general approach and some concrete algorithms. In: Edward Carlstein, Hans-Georg Müller, and David Siegmund (eds.), *Change-point problems*. Institute of Mathematical Statistics, Hayward, CA, 1994, 99-107.

[Dey et al., 2009] Sandipan Dey, Vandana P. Janeja, and Aryya Gangopadhyay. 2009. Temporal neighborhood discovery using Markov models. In: *2009 Ninth IEEE Conference on Data Mining*, 110-119.

[Dong et al., 2012] Wei Dong, Minyi Li, Quoc Bao Vo, and Hai L. Vu. 2012. Reliability in stochastic time-dependent traffic networks with correlated link travel times. In: *2012 15th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 1626-1631.

[GTFS] General Transit Feed Specification Reference. URL https://developers.google.com/transit/gtfs/reference [Accessed 4th December, 2017]

[Hill, 2014] Graeme Hill. 2014. Squeezing the transport data: extracting utility and identifying opportunities. In: *Data Analytics 2014: The Rising Role of Big Data*, 1-22.

[Hounsell et al., 2012] N.B. Hounsell, B.P. Shrestha, and C. D'Souza. 2012. Using automatic vehicle location (AVL) data for evaluation of bus priority at traffic signals. In: *Proceedings of the IET and ITS Conference on Road Transport Information and Control (RTIC 2012)*, 1-6.

[Hu and Hellendoorn, 2013] Yu Hu and J. Hellendoorn. 2013. Uncertainty modeling for urban traffic model predictive control based on urban patterns. In: *2013 16th International IEEE Conference on Intelligent Transportation Systems (ITSC)*, 845-850.

[Jones et al., 2013] Michael Jones, Yanfeng Geng, Daniel Nikovski, and Takahisa Hirata. 2013. Predicting link travel times from floating car data. In: *Proceedings of the 16th International IEEE Annual Conference on Intelligent Transportation Systems (ITSC 2013)*, 1756-1763.

[J_API] Journeys API. URL http://wiki.itsfactory.fi/index.php/Journeys_API [Accessed 4th December, 2017]

[Kafka] Apache Kafka. URL http://kafka.apache.org [Accessed 4th December, 2017]

[Khan et al., 2016] Naveed Khan, Sally McClean, Shuai Zhang, and Chris Nugent. 2016. Using genetic algorithms for optimal change point detection in activity monitoring. In: *2016 IEEE 29th International Symposium on Computer-Based Medical Systems (CBMS)*, 318-323.

[Lau and Ismail, 2015] Sian Lun Lau and S.M. Sabri Ismail. 2015. Towards a real-time public transport data framework using crowd-sourced passenger contributed data. In: *2015 IEEE 82nd Vehicular Technology Conference*, 1-6.

[Mann and Whitney, 1947] Henry B. Mann and Donald R. Whitney. 1947. On a test of whether one of two random variables is stochastically larger than the other. In: *Annals of Mathematics*, 18 (1), 50-60.

[Mu et al., 2010] Beipeng Mu, Jianming Hu, Tingting Zhao, and Yi Zhang. 2010. Evaluating the performance of link travel time estimation based on floating car data. In: *Proceedings of the 2010 International Conference on Optoelectronics and Image Processing (ICOIP)*, 504-508.

[Liang and Ling-xiang, 2006] Zou Liang and Zhu Ling-xiang. 2006. Link travel time estimation model fusing data from mobile and stationary detector based on BP neural network. In: *2006 International Conference on Communications, Circuits and Systems Proceedings*, 2146-2149.

[Pinelli et al., 2015] Fabio Pinelli, Francesco Calabrese, and Eric Bouillet. 2015. A methodology for denoising and generating bus infrastructure data. In: *IEEE Transactions on Intelligent Transportation Systems*, 16 (2), 1042-1047.

[Reynolds, 1994] P.S. Reynolds. Time-series analyses of beaver body temperatures. In: N. Lange, L. Ryan, L. Billard, D. R. Brillinger, L. Conquest, and J. Greenhouse (eds.), *Case Studies in Biometry*. New York: John Wiley and Sons, 1994, 211-228.

[Shewhart, 1931] Walter A. Shewhart. *Economic Control of Quality of Manufactured Product*. D. Van Nostrand Company, Inc., 1931.

[Shimizu et al., 2010] Hikaru Shimizu, Yoshiyuki Moritou, and Masa-aki Kobayashi. 2010. An estimation of link travel time in urban road networks. In: *Proceedings of SICE Annual Conference 2010*, 2901-2905.

[Sunehag et al., 2012] Peter Sunehag, Wen Shao, and Marcus Hutter. 2012. Coding of nonstationary sources as a foundation for detecting change points and outliers in binary time-series. In: *Proceedings of the Tenth Australasian Data Mining Conference*, 134, 79-84.

[Syrjärinne, 2016] Paula Syrjärinne. 2016. *Urban Traffic Analysis with Bus Location Data*. Ph. D. Dissertation, School of Information Sciences, University of Tampere.

[Syrjärinne and Nummenmaa, 2015] Paula Syrjärinne and Jyrki Nummenmaa. 2015. Improving usability of open public transportation data. In: *Proceedings of the 22nd ITS World Congress*.

[Syrjärinne and Nummenmaa, 2016] Paula Syrjärinne and Jyrki Nummenmaa. 2016. Incident detection based on bus data. In: *Proceedings of the 11th ITS European Congress.*

[Syrjärinne et al., 2015] Paula Syrjärinne, Jyrki Nummenmaa, Peter Thanisch, Riitta Kerminen, and Esa Hakulinen. 2015. Analyzing traffic fluency from bus data. In: *IET Intelligent Transport Systems*, 9 (6), 566-572.

[Takayasu, 2015]. Hideki Takayasu. 2015. Basic methods of change-point detection of financial fluctuations. In: *2015 International Conference on Noise and Fluctuations (ICNF)*, 1-3.

[Takeuchi and Yamanishi, 2006] Jun-ichi Takeuchi and Kenji Yamanishi. 2006. A unifying framework for detecting outliers and change points from time series. In: *IEEE Transactions on Knowledge and Data Engineering*, 18 (4), 482-492.

[Taylor, 2000] Wayne A. Taylor. 2000. Change-point analysis: A powerful new tool for detecting changes. URL http://www.variation.com/cpa/tech/changepoint.html [Accessed 4th December, 2017]

[Ye et al., 2016] Hanjie Ye, Jianming Hu, Xudong Xie and Yi Zhang. 2016. Traffic detectors' deployment modeling and optimization under urban road network. In: *2016 12th World Congress on Intelligent Control and Automation (WCICA)*, 2705-2710.

[Zambrano et al., 2016] Jorge L. Zambrano, Carlos T. Calafate, David Soler, Juan-Carlos Cano, and Pietro Manzoni. 2016. Using real traffic data for ITS simulation: procedure and validation. In: *2016 Intl IEEE Conferences on Ubiquitous Intelligence & Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress*, 161-170.