Automated offline detection of disruptions using smart card data

A case study of the metro network of Washington DC

F.Z.G. Jasperse

TU Delft

Automated offline detection of disruptions using smart card data

A case study of the metro network of Washington

by

F.Z.G. Jasperse

to obtain the degree of Master of Science
at the Delft University of Technology,
to be defended publicly on Thursday January 30, 2020 at 11:00 AM.

Student number:	4168739	
Thesis committee:	Dr. O. Cats	TU Delft
	Dr. M. Snelder	TU Delft
	Dr. Y. Huang	TU Delft
	P.K. Krishnakumari	TU Delft

TU Delft

Preface

Dear reader,

This thesis is my final product of the Transport, Infrastructure and Logistics master at the TU Delft. It describes a method to automatically detect disruptions in metro networks, using smart card data, and its results when applied to the case study of the metro network of Washington. The smart card data of this network have been made available by the Washington Metropolitan Area Transit Authority.

First of all, I would like to express my gratitude to my supervising committee. Maaike Snelder, thank you for your excellent guidance and always useful and positive feedback during our meetings. Panchamy Krishnakumari, thank you for providing me with the processed data, always helping me out with any programming issues and giving me inspiration to continue when I was stuck. Oded Cats, thank you for providing me with your extensive and helpful feedback. Yilin Huang, thank you for your useful insights and feedback from another angle.

I would also like to thank my parents, Jacco and Inge, and my friends for being there for me and distracting me when I needed it. Finally, I would like to thank Matthijs for always supporting me and helping me focus when I needed it.

Faye Jasperse
Delft, January 2020

Contents

List of Figures

List of Tables

1 Introduction

In this chapter, the context of this research is introduced. First, some background information on disruption detection and its importance is given in section 1.1. After that, the problem context and relevance of research are described in section 1.2, followed by an introduction to the case study in section 1.3 and the related work including the research gap in section 1.4. This is followed by section 1.5, which introduces the main research question and sub-questions. The chapter is concluded by section 1.6, which gives an outline of the structure of the report.

1.1. Background

Service reliability is one of the most critical performance measures for public transport users. It affects both their perception of service quality as well as their travel behaviour. Transport operators can only understand and improve reliability if they can measure it. Previously, quantifying the reliability of the network was done by conducting surveys or using data from a supply perspective, such as automatic vehicle location (AVL) data, to represent passenger experiences. With the uprise of data collection using automated fare collection (AFC), travel times experienced by passengers can be directly observed and therefore improved estimates of the reliability of the public transport service can be obtained (Uniman, 2009). One of the main interests of public transport operators is to ensure that the quality of service of the network remains competitive from a passenger's perspective when compared to other modes (Kittelson & Associates, et al., 2003). Automated offline disruption detection can help to measure and improve the reliability of public transport networks. It provides transport operators with the duration of disruptions and which locations are most often affected and therefore supports them to prioritise what areas to focus on to reduce passenger delays. Furthermore, smart card data is an interesting source to potentially detect disruptions, as it makes it possible to quantify delays on a passenger level.

1.2. Problem context and relevance of research

Disruptions in public transport can have a massive impact on passenger delays. Due to these delays, passengers may miss their connection, which leads to more delay than the delay of an individual train (Jespersen-Groth et al., 2009). Passengers expect to arrive according to the timetable, and disruptions can lead to a decrease in the network's capacity, which in turn leads to unsatisfied demand and delays (Kepaptsoglou and Karlaftis, 2010). Disruptions in public transport can, for example, be caused by infrastructure malfunctions, rolling stock break downs and accidents (Jespersen-Groth et al., 2009). According to Ji et al. (2018), early detection of disruptions on public transport networks is essential to reduce their impact.

Data used to detect these disruptions are for example reports from transit operation patrols, calls from passengers, traffic sensors and camera monitoring (Hemminki et al., 2013)(Jiang et al., 2011) or WiFi data (Ji et al., 2018). The major drawbacks of these methods are that they are either not precise (such as the reports from transit operation patrols and calls from passengers) or that the method does not cover the whole area (such as the traffic sensors and camera monitoring) (Ji et al., 2018). Furthermore, WiFi data are difficult to use because the frequency of sending probe requests varies a lot and depends on the operating system of the phone as well as on the power state of the phone (on or stand-by) (Song and Wynter, 2017).

Automated disruption detection provides insight into the number of disruptions that occur and their impact. Public transport operators can only understand and improve the reliability if they are able to measure it. Automated disruption detection can help public transport operators to monitor their service performance, therefore this can aid in improving the reliability and reducing passenger delays. It provides them with the duration of disruptions and which locations are most often affected, and the related passenger delays, and therefore supports them to prioritise what locations to focus on to reduce passenger delays. This is also of financial importance: many operators currently offer delay compensations (Krishnakumari et al., 2019). Furthermore, this research can be used as a step towards real-time disruption detection.

1.3. Case study context

In this research, the metro network of the Washington Metropolitan Area Transit Authority (WMATA) will be used as a case study to investigate how disruptions can be detected automatically. The metro network of WMATA has a length of 117 miles (about 188 kilometres) with 95 stations (WMATA, 2019a). Including the bus network, it serves a population of approximately 4 million people (WMATA, 2019a). When a disruption occurs, WMATA can use gap trains as a replacement for train failures (Pender et al., 2012). A gap train is a train that can be inserted on the track when a disruption occurs (Fox, 2018). Furthermore, according to Pender et al. (2012), WMATA has 25 buses that are primarily used for regular bus operations. However, these buses can also be called upon when disruptions in the metro network occur. Still, using buses to replace disrupted metro trains should be minimised as this increases the number of transfers (Pender et al., 2012).

In 2018, WMATA implemented the so-called rush hour promise, which meant that when a trip was delayed by 15 minutes or more, the trip costs were refunded (WMATA, 2019b). In this year, around 0.3% of all rush-hour trips had a delay of 15 minutes or more; this has cost Metro approximately 1 million dollars. However, in 2019, the goal to be on time will be even harder to meet, as the delay of 15 minutes has been decreased to 10 minutes (WMATA, 2019b). Therefore, WMATA needs to detect disruptions efficiently and determine their impacts. This can be accomplished by analysing smart card data, based on tap-in and tap-out records.

1.4. Related work and research gap

Some research has already been carried out on the topic of disruption detection. First, research on disruption detection in public transport networks will be shortly discussed, followed by research on disruption detection in a road network. A more elaborate literature review can be found in Chapter 2. Regarding disruption detection in public transport networks, Ji et al. (2018) used Twitter data to detect metro service disruptions in the WMATA network in real-time. They investigated if service failures cause travel delays for passengers. The advantages of using Twitter data are that it is prompt and geolocated. However, Twitter data might not always be reliable, and it cannot be used to determine the impact by counting the number of passengers that are affected by disruptions. Another way of real-time monitoring public transport systems can be done by using WiFi analytics (Song and Wynter, 2017). This can be used to determine the actual timetable of a public transport system in real-time, instead of the published timetable. Another research, related to predicting disruptions in public transport networks, carried out by Noursalehi and Koutsopoulos (2016), used smart card data to predict arrivals at each station, the origin and destination of passengers and the load for each train. They developed a hybrid data- and model-driven decision support system. This model could be applied to proactive crowd management, by closing gates at stations upstream the line instead of closing the gates at the busy stations themselves. Furthermore, this model could be used to inform passengers real-time about gate closures and service disruptions. However, they did not detect the disruptions itself; planned and unplanned disruptions entered the model as explanatory variables, as soon as they were known. Briand et al. (2018) developed a method to identify disruptions in public transport, considering the changes in transport demand. In this research, a disruption is determined as an unexpected (regarding the circumstances) increase or decrease of tap-in and tap-out data, compared to registrations under normal conditions. Their main goal was to build a robust detection methodology that can detect abnormal transport demand, taking into account the circumstances such as the day of the week, school holidays, school or university peak hours etcetera. Days with a similar activity profile were clustered. However, their analysis was based on demand patterns rather than on the delays. Regarding disruption detection in road networks, Lopez et al. (2017) used clustering to create 3D speed maps, to investigate the regularity of urban congestion patterns. They designed a real-time method to predict travel times. In the literature review in chapter 2, the literature previously mentioned in this section will be elaborated upon. To the author's knowledge, automated offline detection of disruptions in metro networks using a data-driven approach with smart card data has not been done before.

1.5. Research questions

The objective of this research is to detect disruptions automatically, to give insight into the impact of disruptions on passenger delays. Therefore, the research question is as follows:

How can disruptions and the related passenger delays in a metro network be detected automatically offline, using smart card data?

The following sub-questions help to answer the main research question:

1. *How can a distinction be made between a regular delay and a disruption?*
2. *How can disruptions be detected, using the distinction between a regular delay and a disruption?*
3. *How can detected disruptions be used to calculate the related passenger delays?*

The first sub-question can be answered by determining what regular delays look like. Then a distinction can be made between regular delays and disruptions. The second sub-question can be solved by looking at the numerical difference between regular delays and the disruption — this way, the irregular delays can be found. The third sub-question can be answered by determining the number of passengers that are affected by the disruption and to what extent of time.

1.6. Outline

Chapter 2 comprises a literature review to explore existing research on the topic of disruption detection. Then, in chapter 3, the methodology that will be used to answer the sub-questions will be described. Chapter 4 will be used to describe the case study the methodology will be applied on. Chapter 5 will present the results. Finally, the report will be concluded by chapter 6 containing the conclusions and recommendations that follow from this research. A paper written about this research can be found in Appendix E.

2

Literature review

In this chapter, research that has already been done on the topic of disruption detection will be discussed. In Table 2.1, an overview can be found of some of the related research that has already been done, including the data that have been used to detect the disruptions. To select literature, keywords related to disruption detection have been used to search online databases. After reading the abstract and a quick scan of the paper, the relevance of the paper was assessed. Furthermore, it was aimed to create a variety of literature based on modality (public transport or car) and data that was used to detect disruptions. The papers that were considered the most relevant will be discussed in this literature review. First, relevant literature that involves disruption detection on roads will be reviewed, in section 2.1. The relevant research regarding disruption detection on roads is performed by Lopez et al. (2017) and Parsa et al. (2019), where the first used speed observations as an input and the second used real-time loop detector data. After that, literature on the topic of disruption detection in public transport using internet data will be reviewed in section 2.2. Song and Wynter (2017) used WiFi data to detect disruptions, while Ji et al. (2018) used Twitter data. This chapter will conclude with a section on literature that involves smart card data to detect disruption, section 2.3. Noursalehi and Koutsopoulos (2016), Briand et al. (2018) and Tonnelier et al. (2018) all used smart card data as an input for their research.

Author	**Year**	**Modality**	**Disruption detection data**
Noursalehi & Koutsopoulos	2016	Public transport	Smart card data
Song & Wynter	2017	Public transport	WiFi data
Lopez, Leclercq, Krishnakumari, Chiabaut & Van Lint	2017	Car	Speed observations
Ji, Fu, Self, Lu & Ramakrishnan	2018	Public transport	Twitter data
Briand, Toqué, Côme & Oukhellou	2018	Public transport	Smart card data
Tonnelier, Baskiotis, Guigue & Gallinari	2018	Public transport	Smart card data
Parsa, Taghipour, Derrible & Mohammadian	2019	Car	Real-time loop detector data

Table 2.1: Selection of literature on the topic of disruption detection

2.1. Disruption detection on roads

Lopez et al. (2017) investigated the daily regularity of urban congestion patterns using 3D speed maps. First, link speed data was partitioned every 10 minutes. Then, 3D speed maps were created for each day by clustering this partitioned link speed data. After that, each partition could be assigned a cluster number. This means that each day could be represented as a vector of all partitions, whose values are the cluster numbers. Finally, four 3D speed maps were created that relate to four groups of days to describe the daily traffic dynamics. These 3D speed maps were used to real-time predict travel times.

Parsa et al. (2019) developed a model to, instead of predicting travel times real-time, detect road accidents real-time. They used a part of the accident data, loop detector data and weather condition data to train the model. The other part of the data was used to test the model. They then created two models: one using support vector machine and the other one using a probabilistic neural network. The first one is a supervised machine learning method mostly used for classification. The latter is a feedforward neural network algorithm

that is mostly used for classification and pattern recognition. To evaluate the performance of the model, the accuracy, detection rate and false alarm rate were calculated for both models. It was concluded that the probabilistic neural network model performed better than the support vector machine model.

2.2. Disruption detection in public transport using internet data

Song and Wynter (2017) used WiFi data to monitor the level of service in a public transport system. The advantage of using WiFi data is that WiFi access points are widely spread, and most mobile devices are currently equipped with WiFi functionality. To connect a mobile device to WiFi, probe requests are sent from this device. WiFi data have already been used to estimate the number of passengers in public transport vehicles (Handte et al., 2014) and to infer the trajectories of vehicles (Musa and Eriksson, 2012). The goal of the research of Song and Wynter (2017) was to determine the real-time train timetable. Therefore, the timestamps at each station of the mobile devices connected to WiFi were clustered and then it was determined which clusters belong to the same train. The resulting real-time table then provides information about the actual train headways and the in-station dwell times.

Ji et al. (2018) used Twitter data for early detection of metro service disruptions. They filtered tweets using names of metro stations and metro lines that are operated by WMATA. After that, the tweets are grouped per metro line. They looked at the spatial connectivity of metro lines as multiple lines may provoke similar complaints in tweets. Then they looked at common complaint vocabulary across metro lines. When a disruption occurs, it is assumed that the vocabulary of tweets is similar across the metro lines. Ji et al. (2018) created a training set and a test set using the disruption log file of WMATA. The results demonstrate that their model can be used to detect metro delays effectively.

2.3. Disruption detection using smart card data

Noursalehi and Koutsopoulos (2016) developed a hybrid data and model-driven decision support system, using smart card data. They created real-time predictive models to help transit operators manage disruptions and respond proactively to them. Automatic Fare Collection (AFC) systems result in useful smart card data comprising an enormous amount of detailed information on the trips passengers are making and their travel behaviour. Noursalehi and Koutsopoulos (2016) developed demand decision support systems and advanced passenger information services using smart card data. This research aimed to predict arrivals at each station, the origin and destination of passengers and the load for each train. They predicted 15 to 30 minutes ahead to be able to create a realistic prediction and at the same time have the opportunity to react on a disruption proactively. In the first step of their method, the model has to be trained. They clustered similar daily arrivals at each station in a group. Exogenous variables, such as weather conditions, major events on that day or station or line closures, are also incorporated. These exogenous variables enter the predictive models for each station as explanatory variables, making it possible to predict the effect of certain events on the demand. An overview of the offline training of the model can be found in Figure 2.1.

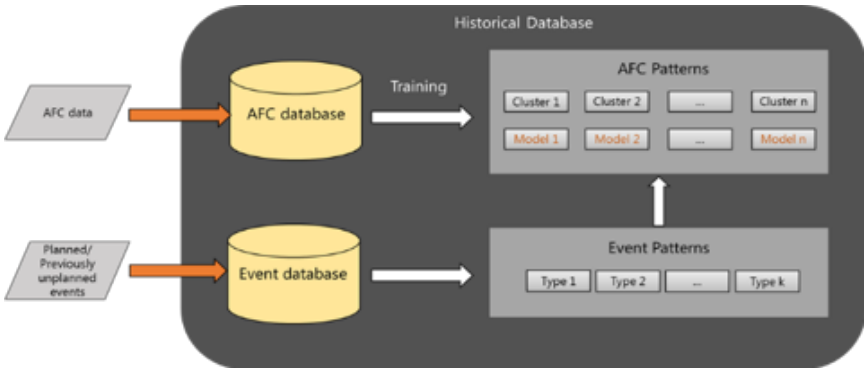


Figure 2.1: Offline training of model (Noursalehi and Koutsopoulos, 2016)

After the offline model was trained, they created a separate predictive model for each group. When the smart card data is being collected, a classifier determines which of the historical patterns is the most similar to the incoming smart card data, and then chooses the corresponding model to predict disruptions. Planned and unplanned events are also used as input to include their effect on the prediction. An overview of the

real-time prediction model can be found in Figure 2.2.

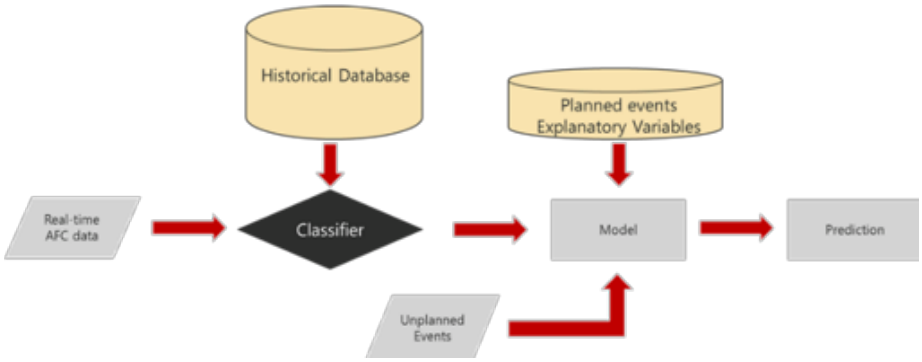


Figure 2.2: Real time disruption prediction (Noursalehi and Koutsopoulos, 2016)

The results of the research of Noursalehi and Koutsopoulos (2016) could be used for proactive crowd management and passenger information services. An often applied crowd management strategy is to close station gates at busy stations and therefore denying the entry of passengers, resulting in long waiting times for a large number of passengers. After the station gates are closed, a part of these passengers move to other nearby stations. These stations are then also at risk to experience a deteriorating performance. Proactive crowd management means that opportunities are identified to close the station gates at stations upstream the line shortly, so crowding at busier stations can be managed. Furthermore, the staff can be alerted earlier, so safety measures can be implemented to manage the increased customer demand. With the improved passenger information services, passengers can be informed about predicted entries at stations and the probability that access will be denied. On top of that, registered passengers can be warned with specific information that their trip might be disrupted and alternative routes could be proposed to them.

Briand et al. (2018) detected unexpected increases or decreases in passenger demand, using smart card data. These demand increases or decreases could be caused by exceptional events, an operational problem such as a breakdown resulting in a smaller or larger number of registered passengers or a problem in the automated fare collection system, resulting in an inaccurate number of ticket validations. The research of Briand et al. (2018) aims to detect outliers in smart card data. First, the types of days at all stations were identified by using a clustering method. Then for each station and days of each type that were identified using the clustering stage, anomaly detection is executed. For the first part, a quarter-hourly validation vector was defined. This vector was defined for each day, containing the aggregated number of ticket validations for each line. Then, clusters were defined by assigning categories to the days such as which weekday and month it is and whether the day is during a public holiday, school holiday or a four-day weekend. Days with the same category are then put into the same group. After that, a median plot is computed for each group, because a median is a robust estimator and not very sensitive to outliers. This median is computed for each quarter-hourly point. Then hierarchical clustering was applied to these median plots to create clusters that contain similar days. After that, two different boxplot methods are used to detect atypical events. The first, classical boxplot method is applied to each quarter-hourly vector. When a vector lies outside the interquartile range of the boxplot, it is considered an atypical event. The second, functional boxplot method is applied to all ticket validations within a day to determine if a day is atypical as a whole. Then, both methods were evaluated using a disruption log file. It was concluded that the functional boxplot method detects more anomalies that are present in the disruption log file than the classical boxplot method. On top of that, the functional boxplot method resulted in a lower false-positive rate than the classical boxplot method.

Tonnelier et al. (2018) presented four approaches to detect disruptions in a transportation network using smart card data. The first approach is to create a baseline model. For each day of the week, the average number of entry-logs per station is calculated. This average activity reference is used to define an anomaly score. This anomaly score considers the difference between the average behaviour of a station and the actual behaviour of a station for a given day. The second approach is the normalised baseline model. This uses the same model as in the first approach but normalises the entry-logs of each station. This results in a probability density function which gives the probability of observing a certain activity at a certain time, given the day and station. The advantage of the normalised baseline model is that it can detect more fine-grained anomalies. The third approach uses the Nonnegative Matrix Factorisation algorithm. This algorithm produces the av-

erage behaviour of a station, divided into a couple of weighted common temporal patterns that are shared by the whole network. This results in a discrete probability density function. This function can then be used to calculate the anomaly score. The final approach is the continuous user-based model. This model uses the habits of the passengers who have entered the station to detect anomalies. This model leads to a more fine-grained approach but is also very computationally intensive.

2.4. Conclusion

To detect disruptions, this research will first investigate the day-to-day regularity of delays, as in Lopez et al. (2017), but then applied to a public transport network instead of a road network. This research will use smart card data as an input, as Briand et al. (2018) also have, but will use delays to identify disruptions, rather than demand patterns. Noursalehi and Koutsopoulos (2016) also used smart card data, but they did not use it to analyse disruptions, as planned and unplanned disruptions entered the model as explanatory variables.

One important conclusion of the literature review is that detecting disruptions should start by defining average behaviour. Only when average or 'normal' behaviour is known, disruptions can be detected. Almost all of the methods used a training set and then applied the outcomes of this training set onto the data. This training set can be used to determine average behaviour. Furthermore, it can be concluded that clustering is a widely used method to detect disruptions. How this method can be applied to this research will be elaborated upon in chapter 3.

3

Methodology

In this chapter, the methods that are used in this study are described. First, the methodology framework will be presented in section 3.1. This section also explains why the different steps in the framework are chosen. The second step, distinguishing between regular and irregular days, will be explained in section 3.2, followed by the third step, detecting disruptions, in section 3.3. This chapter will conclude with a section on verification, section 3.4.

3.1. Methodology framework

First, smart card data is used to calculate the average passenger delay at each station, track segment and interchange station in the network. In the context of this study, a disruption is defined as a delay that does not occur regularly in the network. To distinguish between a delay that occurs regularly and a disruption, the day-to-day regularity of delays must be investigated. When delays fit a similar pattern, for example, when a delay regularly occurs at the morning peak, the delays belonging to this pattern are not considered to be a disruption. To investigate the day-to-day regularity, days can be clustered. As can be seen in Figure 3.1, splitting the data set into training data and test data constitutes the first step in the methodology framework. The training data is used to perform the clustering in the second step. This second step constitutes of distinguishing between regular and irregular days. The output of this step are days that have irregular delay patterns; these are the days that potentially contain disruptions. The third step is to detect the disruptions within these irregular days. Finally, areas in the network with dissimilar delay patterns are identified, the disruptions. After detecting the disruptions, these disruptions need to be verified. This can for example be done by comparing the disruptions that are detected by the algorithm to other available data, such as train movement data. When it is known that a train was standing still for a longer time than scheduled, it is expected that a disruption might occur.

3.2. Distinguishing between regular and irregular days

To divide days into regular and irregular days, the days can be clustered. Clustering is used to group similar objects, in this case days. To be able to use the clusters as a training set, only one part of the data has to be clustered. These clusters will then be used to create a probabilistic classifier, which is in turn applied to the other part of the data, the test set. This probabilistic classifier is used to determine for each of the days in the test set, the probability that it belongs to each of the created clusters using the training set. If it has a low probability of belonging to any of the clusters, or a high probability of belonging to a cluster that represents irregular days, it is considered to be an irregular day. This day will then be used in the next step to detect disruptions within this irregular day. The two main methods in this step, clustering and probabilistic classification, will be explained in the following sections.

3.2.1. Clustering

Two of the most used clustering methods are K-means clustering and hierarchical clustering (Sharma, 2018). These two methods are described in the next part.

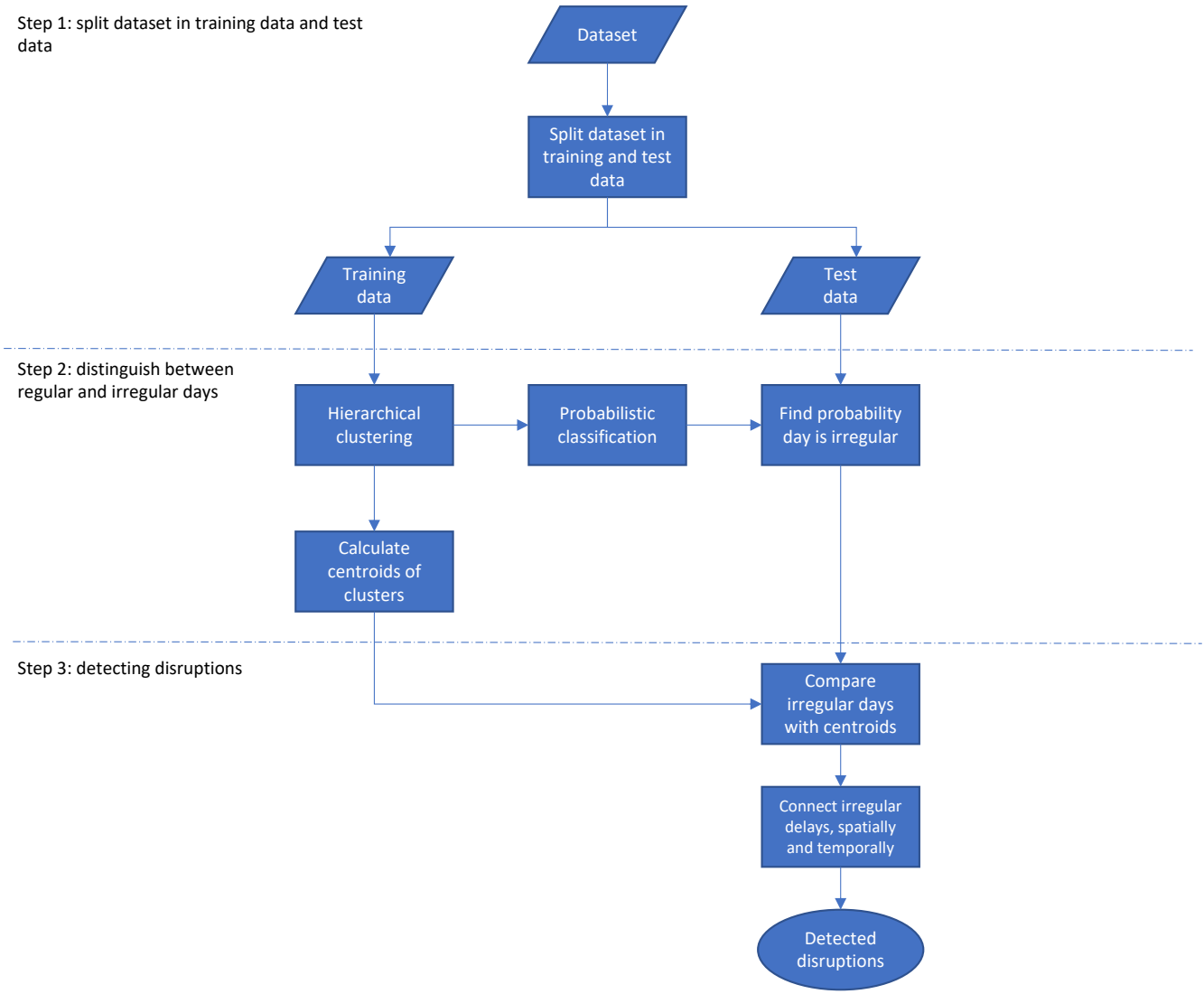


Figure 3.1: Methodology framework

K-means clustering

The aim of K-means clustering is to divide M points in N dimensions into K clusters, such that the sum-of-squares within a cluster is minimised (Hartigan and Wong, 1979). The inputs for this algorithm are a matrix of M points in N dimensions and a matrix of K initial cluster centers in N dimensions. To determine the best partitioning in clusters, the error of the clustering has to be calculated. This error e is first calculated for the initial partitioning. It consists of the sum of squared distances of the data points from their cluster means. The basic idea of K-means clustering is to search for a partitioning with a small e by moving data points from one cluster to another until no such movement reduces e (Hartigan, 1975). A disadvantage of K-means clustering is that the amount of clusters K has to be known beforehand. Therefore, another method will be looked at in the next section, hierarchical clustering.

Hierarchical clustering

Hierarchical clustering is a clustering method that does not induce a single clustering but creates a hierarchical structure of clusters. The two main hierarchical clustering algorithms are an agglomerative and a divisive one. In the agglomerative (or bottom-up) algorithm, each data point N is initially a separate cluster, after which pairs of clusters are merged until all data points belong to a cluster. This means that the first partition consists of N clusters and the last consists of one single cluster with all N data points. The divisive (or

top-down) algorithm is precisely the opposite: initially, all data points belong to a single cluster, and these are split up until each data point belongs to a different cluster. This means that the first partition consists of only one cluster and the final partition has N clusters (Camargos et al., 2016). The agglomerative hierarchical clustering is less complex and more widely used and is therefore the algorithm that will be used in this study. Different methods can be used to define the inter-cluster distance. The three most common ones are single linkage, complete linkage and Ward linkage (Shalizi, 2009).

With single linkage, the distance between two clusters is defined as the distance of the closest pair of data points, where only pairs that consist of one data point from each cluster are considered. This is also known as the nearest point algorithm. The inter-cluster distance can be calculated using single linkage with the formula in equation 3.1.

$$d(u,v) = \min(dist(u[i], v[j])), \tag{3.1}$$

for all points i in cluster u and j in cluster v (SciPy, 2019). A major drawback of this method is that it can create long thin clusters, in which the data points of one cluster are close to each other, but the data points at the end of the cluster are far away from each other. It only controls the nearest neighbours similarity. Therefore, using single linkage to cluster the days is probably not the most useful method.

With complete linkage, the distance between clusters is defined as the distance between the farthest two data points, one of each cluster (Shalizi, 2009). This can be calculated with the formula in equation 3.2.

$$d(u,v) = \max(dist(u[i], v[j])) \tag{3.2}$$

for all points i in cluster u and j in cluster v (SciPy, 2019).

The complete linkage method avoids the drawback of the single linkage method that can create long thin clusters, and can instead find more compact clusters. However, outlying data points prevent otherwise close clusters to merge, as the measure of the furthest, outlying data point avoids the two clusters to merge (Yim and Ramdeen, 2015). In the case of clustering days based on the delays during those days, it might be the case that a day with very large delays avoids two close clusters to merge. The complete linkage method is therefore probably not the most useful in this case.

With Ward linkage, the inter-cluster distance is defined as how much the sum of squares will increase when two clusters are merged (Shalizi, 2009). The formula that can be used to calculate this distance can be found in equation 3.3.

$$d(u,v) = \sqrt{\frac{|v|+|s|}{T}d(v,s)^2 + \frac{|v|+|t|}{T}d(v,t)^2 - \frac{|v|}{T}d(s,t)^2} \tag{3.3}$$

where u is the newly joined cluster consisting of clusters s and t, v is an unused cluster in the forest, $T = |v| + |s| + |t|$ and $|*|$ is the cardinality of its argument (SciPy, 2019). The cardinality is the number of elements in the set. With agglomerative hierarchical clustering, the sum of squares is initially zero, because every data point is in a separate cluster and then increases as clusters are merged. Ward's linkage method aims at keeping this growth as small as possible (Shalizi, 2009). Because Ward's method takes all data points of a cluster into account instead of just the two closest or most distant data points of two clusters (as in the single and complete linkage), Ward's method is less susceptible to outliers (Jobson, 1992). A possible problem when clustering days based on the delays, when the aim is to create clusters that present 'regular' days, is that irregular days have a large influence on the clusters. As outliers (or irregular days) have less impact on the clustering of Ward's method, this might be a good method to use.

To determine whether Ward's method is indeed the most suitable method to cluster the days, the three different methods will be applied to the data. The results can be found in chapter 5, section 5.1.

When compared to K-means clustering, hierarchical clustering does not need a specified number of clusters beforehand; the cluster merging can be visualised in a dendrogram which can be used to decide upon the number of clusters. An example of such a dendrogram can be found in Figure 3.2.

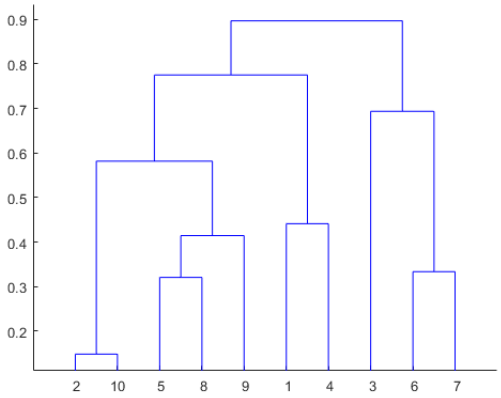


Figure 3.2: Example of a dendrogram (Mathworks, nd)

The horizontal lines are where the clusters merge, the heights of these lines show the distance between the clusters that are merged. The vertical lines show which clusters were part of the merge that forms that new cluster.

Except that the number of clusters cannot be decided upon after visualising the cluster merging, another disadvantage of K-means clustering is that the initial centroids (the number of centroids is equal to the number of clusters) need to be chosen, which is done by selecting random data points when using the traditional K-means clustering (Baswade and Nalwade, 2013). This random selection of initial centroids can influence the quality of the clustering. Hierarchical clustering does not need to deal with initialising centroids. To summarise why hierarchical clustering is chosen: hierarchical clustering does not require the number of clusters beforehand. Instead, the cluster merging can be performed using a dendrogram, and it does not require a pre-selection of initial centroids. Because of these advantages of hierarchical clustering over K-means clustering, hierarchical clustering will be used in this research.

The hierarchical clustering described above solely uses data point values and the distances between days to cluster. However, it might be interesting to also look at the similarity indices between days. This method will be described in the next section.

Clustering based on similarity matrix indices

The Structural SIMilarity (SSIM) index is a method to measure the similarity between two images (Wang et al., 2004). However, this index can also be applied to compare OD-matrices (Origin and Destination matrices). Djukic et al. (2013) calculated the structural similarity indices between OD matrices to compute the quality of estimated OD matrices. Most statistical measures that are used to assess the quality of estimated OD matrices only compute the difference between an estimated OD matrix and the reference OD matrix. These statistical measures do not take the spatial and temporal correlation in OD matrix data into account. The delays in this research are also spatially and temporarily correlated as delays spread over the network in both space and time. Therefore, the SSIM index can be a useful measure to compare days, also structurally. Djukic et al. (2013) represented OD matrices as images, where the origin zones were given in rows and the destinations in columns. The number of trips per OD pair was used to create a colour map to determine the colour for each OD pair. This can also be applied to the days that will be compared in this research. In this case, the links, node stations and transfer stations could be presented on the x-axis, the timestamps of half an hour on the y-axis and the delay in minutes can be used to create a colour map to show the delays in colour. An example of such an image of a day can be seen in Figure 3.3. In this image, the first 193 data points on the x-axis represent the average link delays per passenger, the second 193 data points represent the average node delays, while the last 193 data points show the average transfer delays. Hence the left third part of the image shows the link delays, the middle part the node delays and the right third part indicates the transfer delays.

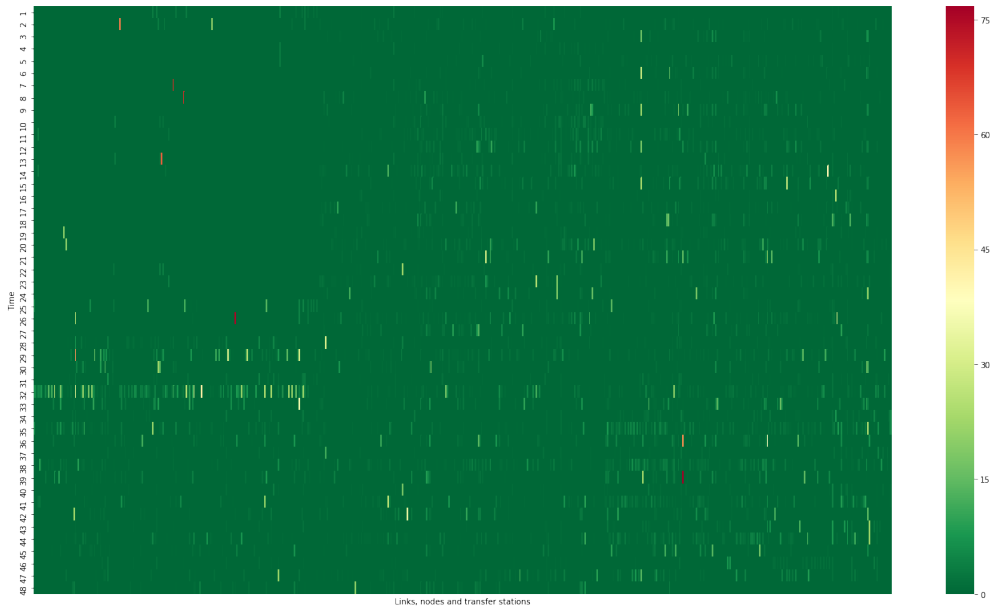


Figure 3.3: Visualisation of link, node and transfer delays per time slice for October 17, 2017

The SSIM index can be calculated with the formula that can be found in equation 3.4.

$$SSIM(d,\hat{d}) = [l(d,\hat{d})^{\alpha}][c(d,\hat{d})^{\beta}][s(d,\hat{d})^{\gamma}] \tag{3.4}$$

In this equation, l represents a distance metric that is used to compare the mean values of the two matrices. c is used to compare the standard deviation of the matrices, while s is used to compare the structure of the matrices.

$l(d,\hat{d})$ is defined by equation 3.5.

$$l(d,\hat{d}) = \frac{2\mu_d\mu_{\hat{d}} + C_1}{\mu_d^2 + \mu_{\hat{d}}^2 + C_1} \tag{3.5}$$

Equation 3.5 is thus used to compare the mean values of vector d and $\hat{d}$ and $\mu_d = \frac{1}{N}\sum_{n=1}^{N} d_n$.

$c(d,\hat{d})$ is written in equation 3.6.

$$c(d,\hat{d}) = \frac{2\sigma_d\sigma_{\hat{d}} + C_2}{\sigma_d^2 + \sigma_{\hat{d}}^2 + C_2} \tag{3.6}$$

Equation 3.6 is used to compare the standard deviation, or square root of variances, of the vectors and $\sigma_d = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(d_n - \mu_d)^2}$.

Finally, the formula to calculate the structure term $s(d,\hat{d})$ can be found in equation 3.7.

$$s(d,\hat{d}) = \frac{\sigma_{d\hat{d}} + C_3}{\sigma_d\sigma_{\hat{d}} + C_3} \tag{3.7}$$

In equation 3.7, $\sigma_{d\hat{d}} = \frac{1}{N-1}\sum_{n=1}^{N}(d_n - \mu_d)(\hat{d}_n - \mu_{\hat{d}})$. Structure term $s(d,\hat{d})$ indicates the similarity between two vectors. It is defined as the correlation between the two vectors and is equivalent to the correlation coefficient that measures the degree of linear correlation. Geometrically, the structure term represents the cosine of the angle between the two vectors, independent of the vector lengths. It is equal to one if, and only if, the structures of the two vectors that are being compared are exactly the same.

Constants C_1, C_2 and C_3 in equations 3.5, 3.6 and 3.7 are used to stabilise the metric when the means and variances would become close to zero. When $C1 = C2 = 0$, unstable results would be produced. The parameters α, β and γ in equation 3.4 are used to adjust the relative importations of the components $l(d,\hat{d})$, $c(d,\hat{d})$ and $s(d,\hat{d})$. According to the founders of the SSIM index, Wang et al. (2004), it is recommended to set $\alpha = \beta = \gamma = 1$ and $C_3 = C_2/2$ to simplify the expression. Equation 3.4 can then be written as in equation 3.8.

$$SSIM(d,\hat{d}) = \frac{(2\mu_d\mu_{\hat{d}} + C_1)(2\sigma_{d\hat{d}} + C_2)}{(\mu_d^2 + \mu_{\hat{d}}^2 + C_2)} \tag{3.8}$$

Combined clustering

To take both the Euclidean distance and the structural similarity index into account, both measures can be combined by applying combined clustering. After the days are clustered based on both the Euclidean distance and the SSIM index, the resulting cluster IDs are used to create new vectors for each day to combine both methods. Consequently, each day is represented by a vector with a length of two: the cluster ID resulting from the clustering based on the Euclidean distance and the cluster ID resulting from the clustering based on the SSIM index. Finally, hierarchical clustering is applied for the third time to these combined vectors. This results in a number of classes that each contains a certain number of days. An overview of this combined clustering can be found in Figure 3.4.

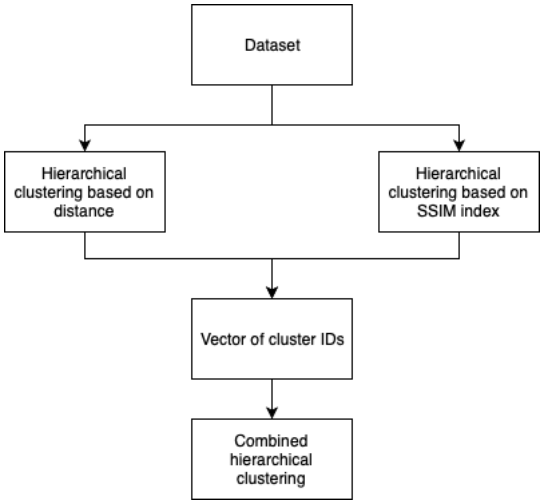


Figure 3.4: Overview of combined clustering

3.2.2. Probabilistic classification

In the previous step, clusters were formed that take both the Euclidean distance as well as the similarity indices between days in account. This was done for the training data. These clusters can now be used to determine the probability of each day belonging to a certain cluster for the days in the last six months. The method that will be used is called probabilistic classification. Probabilistic classification differs from regular classification in the sense that it does not only provide a guess at the class label but calculates the probability of belonging to each class (Rasmussen and Williams, 2006). To discuss approaches to classification, it is needed to look at the joint probability $p(y,x)$, where y stands for the class label, and x denotes the input pattern. When Bayes' theorem is used, this joint probability can be split as either $p(y)p(x|y)$ or as $p(x)p(y|x)$. Therefore, two approaches can be distinguished to solve classification problems. The first method is called the generative approach. This approach models the class-conditional distributions $p(x|y)$ for $y = C_1, ..., C_C$ and the prior probabilities of each class. After that, it computes the posterior probability for each class using the formula in equation 3.9.

$$p(y|x) = \frac{p(y)p(x|y)}{\sum_{c=1}^{C} p(C_c)p(x|C_c)} \tag{3.9}$$

The second approach is called the discriminative approach. This approach models $p(y|x)$ directly. Choosing one of the two approaches can be a difficult decision. The advantage of the discriminative approach is that it models the desired outcome, $p(y|x)$ directly. Furthermore, probabilistic classification is complex when x is high dimensional. This means that if classification is the only desired result, the generative approach might be more complex than necessary, and therefore the discriminative approach may be preferred. However, when missing input values, outliers and unlabelled data points are present in the data, it is useful to determine $p(x)$, which can be obtained by calculating class label y from the joint probability as $p(x) =$

$\sum_y p(y)p(x|y)$ in the generative approach (Rasmussen and Williams, 2006). In this research, x is high dimensional, and there are no missing input values or unlabelled data and very few outliers. Therefore, it is chosen to use a discriminative approach, as it is also less complex. One of the most used discriminative approaches is the linear logistic regression model.

Linear logistic regression model

The linear logistic regression model calculates the probability that x belongs to a certain class as in equation 3.10 (Rasmussen and Williams, 2006).

$$p(C_1|x) = \lambda(x^\top w), \ \ where \ \ \lambda(x^\top w) = \frac{1}{1+e^{(x^\top w)}} \tag{3.10}$$

for a certain weight vector w. The traditional linear logistic regression model is used for binary classification only. When modelling for more than two classes, a multinomial linear logistic regression model is needed (Starkweather and Moske, 2011). Multinomial logistic regression has the advantage that normality, linearity or homoscedasticity are not assumed. However, it does assume that the membership of a certain class is not related to the membership of another class. It also assumes non-perfect separation: if the classes are perfectly separated, unrealistic coefficients will be estimated. With real data, it is unlikely that the data can be exactly separated (Hladká and Holub, 2018).

3.3. Detection of disruptions

Using the probabilistic classification, the probability of belonging to each of the clusters can be calculated for each day in test data. When the probability of belonging to each of the clusters is lower than a certain threshold, the day is considered to be an irregular day. To detect the disruptions, the irregular day can be compared to the centroid of the cluster with the highest probability. The centroid a vector, representing the means of all data points in a cluster.

3.3.1. Compare irregular days with centroids

To compare the irregular day with the centroids, the centroid that represents the 'regular' delays can be subtracted from the irregular day, so the disruptions remain. To reduce the sensitivity of the algorithm, a threshold for a minimum delay might be used to define a disruption. The centroid that will be used to compare is the one that corresponds to a regular cluster with the highest probability for that specific irregular day. The centroid is a vector that consists of all mean values of each feature of each data point of the vectors belonging to the corresponding cluster. The difference between the centroid of the regular cluster with the highest probability and the irregular day consists of the disruptions, which involve the delays that do not occur regularly.

3.3.2. Detect connected delays

One disruption might impact more than one node, link or transfer station and spread over the network. First, the spatially connected disruptions are detected, to see if a disruption has spread over the network. This can be done by creating a graph of the network. The graph of the network can be written as $G = (V, E)$, where V is a set of vertices (nodes) and E is a set of edges (links). All links are added separately for each direction, and each station is split up in nodes, one related to each outgoing link. For example, when one station has three outgoing links, the station is split up into 3 nodes. Therefore the number of nodes is equal to the number of links. After that, links are added to connect the nodes related to the same station to make a connected graph, but these links are only used to connect the graph and are not relevant for the rest of the analysis as no delay is measured on these links. This means that $V \subset E$. When locations are spatially connected, they are merged into one disruption. When the locations cannot be merged, this means that more disruptions have occurred during the same time slice, in different parts of the network. The algorithm to spatially connect disruptions can be found in Algorithm 1.

Algorithm 1 Spatial connectivity

```
for t ∈ T do                                                    ▷ The number of time slices
    initialise disruption list, exclusion list, neighbour list = 0
    initialise affected locations at time t
    while exclusion list != affected locations do               ▷ While not all locations have been checked
        for i ∈ affected locations do
            if affected location(i) ∉ exclusion list then
                if affected location(i) ∉ disruption list then
                    add affected location to disruption list
                add affected location to exclusion list
                for j ∈ affected locations do
                    if shortest path length from affected locations(i) to affected locations(j) = 1 then
                        add affected location(j) to neighbour list
                for k ∈ neighbour list do
                    if neighbour ∉ exclusion list then
                        add neighbour to disruption list
                if disruption list ⊆ exclusion list then
                    add disruption list to disruption list per time t
```

At the beginning of investigating each time slice, the list of disruptions, the exclusion list and neighbour list are set at 0. The affected locations at that time slice are detected. While not all locations have been checked, it must be checked if the affected location is not in the exclusion or disruption list yet. If it is not, the location is added to the disruption list, and after that, it is added to the exclusion list as this location does not need to be rechecked. Then, for all other locations at that time slice, it is checked if they are spatially connected by calculating the shortest path length between them. If this is equal to 1, the locations are spatially connected, and therefore the location is added to the neighbour list. If this neighbour is not yet in the exclusion list, it is added to the disruption list. If the disruption list is a subset of the exclusion list, all disruptions are checked, and therefore the disruption list is added to the total disruption list per time slice, and the algorithm starts again at the successive time slice. After finding disruptions that are spatially connected, it needs to be checked if the disruptions are also connected temporally. This is done using Algorithm 2.

Algorithm 2 Temporal connectivity

```
for i ∈ NTC-disruptions do                                      ▷ The number of non-temporally connected disruptions
    initialise neighbour list, affected locations = 0
    initialise affected locations = NTC-disruptions(i)
    for k ∈ affected locations do
        for l ∈ number of stations do
            if shortest path length from affected location(k) to station(l) = 1 then
                add station to neighbour list
    for j ∈ NTC-disruptions do
        if timeslice(j) - timeslice(i) = 1 then                 ▷ The disruptions are temporally connected
            if NTC-disruptions(j) ⊆ neighbourlist or NTC-disruptions(j) ⊆ affected locations then
                disruptionID(j) = disruptionID(i)
            if disruptionID(i) = {} then
                disruptionID(i) = max(disruptionID)+1
```

For each disruption that is not yet temporally connected, so the output of Algorithm 1, the neighbour list and affected locations are initialised at 0. The affected locations are initialised at the non-temporally connected locations at one time slice. For each of these locations, the connected locations (stations) are found by calculating the shortest path length between these locations and stations. When this is 1, the locations are neighbours. Then, these stations are added to the neighbour list. For all non-temporally connected disruptions, the difference between time slices is calculated. When this is exactly 1, the time slices in which these disruptions occur are successive. If one of the disruptions of the latest time slice is part of the neighbour list or equal to the affected locations at the previous time slice, the disruptions are temporally connected.

This means that the disruption ID of the previous disruption is assigned to the disruption of the latest time slice. When this is not the case, the disruption ID of the previous time slice will remain empty and is then assigned the maximum disruption ID plus 1. This algorithm results in disruptions that are both spatially and temporally connected.

3.3.3. Calculating passenger delays and other disruption related statistics

After detecting the connected delays, the locations that are affected by delays caused by disruptions are known. This information can be used to create the final output, that contains for each day the number of disruptions, the duration of these disruptions, the spatial extent of each disruption, the average passenger delay and the total passenger delay. The average passenger delay can be defined as the delay experienced by one passenger while travelling over a certain link or node (Krishnakumari et al., 2019). The average passenger delay of a certain disruption is then the sum of the average passenger delay for each node and link that has been affected by the same disruption. The total passenger delay is the total delay incurred by all the passengers that travel over a certain link or node. The total passenger delay can be calculated by multiplying the average passenger delay with the passenger counts. So the total passenger delay can be written as a function of the number of passengers that travel over a certain node or link during a given time period, in this case, the duration of the disruption.

3.4. Verification of detected disruptions

After the disruptions are detected, they should be verified. Because detecting the disruptions is done offline, the disruptions that are detected by the algorithm could be verified by comparing them to historical data of disruptions in the network. This is not completely accurate, as not all delays are caused by disruptions, but it does give an idea on the accuracy of the disruptions that are detected by the algorithm.

4

Case study

In this research, the metro network of WMATA will be used as a case study. This metro network in the metropolitan of Washington is the third largest heavy rail transit system with a track length of 118 miles (190 kilometres), 6 rail lines, 91 stations and 1144 railcars. In 2018, 174 million trips were made with WMATA (WMATA, 2019c). Nearly 9 out of 10 trips arrived on time. To use the metro network, a smart card (SmarTrip) is required. An overview of the metro network can be found in Figure 4.1. This chapter will start with section 4.1, which describes the available data. After that, section 4.2 explains how the smart card data have been processed to calculate average passenger delays. The chapter will conclude with a section describing the input data used for this study, section 4.3.

Figure 4.1: WMATA metro network (WMATA, 2019d)

4.1. Available data

The data that will be used in this study have been made available by WMATA. This data consist of smart card data containing the tap-in and tap-out time stamps and a disruption log file.

4.1.1. Smart card data

The smart card data that are used to calculate the average passenger delays consist of all tap-in and tap-out time stamps of passengers travelling between August 2017 and August 2018. As a smart card is required to use the metro network, no trips are missing from these data. Furthermore, the passenger counts have been calculated, using these smart card data as an input. The smart card data have been analysed to calculate the average delay per passenger for each node, link and transfer station. The method that is used to accomplish this will be explained in section 4.2.

4.1.2. Disruption log file

The disruption log file contains all disruptions occurring between August 2017 and August 2018. This file will be used to verify the disruptions detected by the algorithm that will be developed in this research.

4.2. Calculation of delays based on smart card data

The research of Krishnakumari et al. (2019) preceeds this study and their results are used as an input. They estimated network passenger delays in the metro network of WMATA using individual trajectories from smart card data. WMATA defines a passenger delay as the observed passenger travel time minus the scheduled travel time. The observed passenger travel time is defined as waiting time plus on-board time plus transfer time. The scheduled travel time is defined as the on-board time plus the headway plus walking time. An overview of the scheduled and observed travel time and passenger delay is shown in Figure 4.2. It shows a passenger waiting at station 2, travelling over a black metro line and transferring to a green metro line.

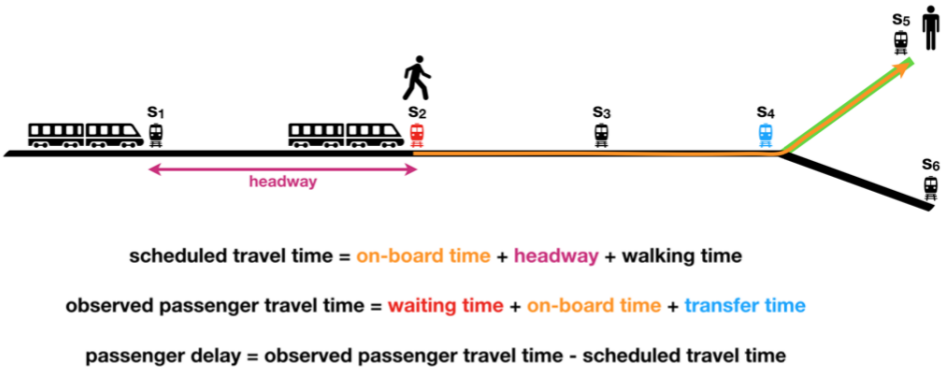


Figure 4.2: Scheduled and observed travel time between station s_2 and s_5 (Krishnakumari et al., 2019)

The delay can be broken down into three parts: delay of the vehicle itself (link delay), delay at the station (node delay) or a transfer delay. First, the network passenger delay estimation problem is described as a solvable set of equations. A directed graph is created to represent the metro network, with the set of stations (nodes), the set of track segments between those stations (links) and the set of ordered pairs representing a metro line and the set of transfer stations where transfers between lines take place. The transfer stations are a subset of the stations. Because the passenger-train assignment has already been carried out, the origin station, destination station and trajectory of each passenger are known. The passenger delay is defined as the the observed travel time minus the maximum scheduled route time (Krishnakumari et al., 2019). When the passenger would be earlier at the destination than expected, the delay is defined to be 0, instead of negative. The observed travel time is obtained by calculating the difference between the tap-in and tap-out time of a specific passenger. The maximum scheduled route time is the sum of the scheduled running and dwell times, the headway between successive trains on a certain route and the walking times at the origin and transfer stations. This is the on-time journey of a passenger. That means that when a passenger just misses a train, but can catch the next train of that line with the scheduled headway, the journey of the passenger is considered to be on-time. Therefore, Krishnakumari et al. (2019) included headways between trains in

calculating the scheduled travel time. The goal is to decompose the delay experienced by the passengers to the corresponding network elements on their route, using the individual trajectories of the passengers. It is assumed that the observed travel time of a passenger consists of three elements:

1. Time spent at the origin stop of the trip
2. Time on-board a train on each of the trip segments
3. Time spent at the transfer stations of the trip

So the delay of a passengers also consists of the delays that occur at these elements. The passenger delay between certain stations for a certain departure time is defined by Krishnakumari et al. (2019) as the initial waiting delay at the origin station plus the on-board delay between each track segment and the transfer delay for each transfer stop on the route of the passenger. This is written as an inequality, so the passenger delay is larger or equal to the sum of the waiting delay, on-board delay and transfer delay, because non-observable personal travel components might add additional delays, for example when a passenger gets a coffee within the gates of the metro station. Using this definition, each passenger trip can be written as a linear combination of all three passenger delay components. These linear combinations can be used to form a potentially solvable system of equations, using as a constraint each passenger trajectory between certain times, for all other trips that use one or more common network elements along this trip. The number of unknown variables is equal to the sum of the number of initial stations, transfer stations and links in the metro network. Krishnakumari et al. (2019) found that an additional constraint can be added to make the system of equations indeed solvable. This constraint means that the on-board delay should be equal to the train delay between the corresponding links. The set of equations is solved with a constrained linear least square solution method, with the lower bound set to 0 so a non-negative solution is ensured. This results in the average passenger delays for each element: the average node delay per passenger for each node (station), the average link delay per passenger for each link (track element) and the average transfer delay for each transfer station. These values have been provided for the purpose of this research by Krishnakumari et al. (2019) and will be used as an input for this study. The time period for which these values have been calculated runs from September 2017 until August 2018.

4.3. Input data used for this study

The average passenger delays calculated by Krishnakumari et al. (2019) will be used as an input for this study. The WMATA network consists of 95 stations. 193 different links can be distinguished when assigning a different number to each link with a different direction. To be able to incorporate direction within the nodes, the nodes are renumbered. One station can get assigned multiple numbers, depending on the link it is connected to. Therefore, also 193 station numbers are distinguished. This is also the case for transfer stations; when a station is not a transfer station, the average passenger transfer delay values at that station will be 0. As the average passenger delay for each track element has been calculated for every half an hour, each day consists of 48 time slices. The data for each day can then be represented as one long vector, with the average passenger node, link and transfer delay for each time slice. Each vector has thus a length of (193*3*48).

5

Results

In this chapter, the results of all the steps in the methodology framework will be shown. First, the three different linkage methods of chapter 3 will be applied to the data in section 5.1. After that, the results of the hierarchical clustering will be shown for week days, first based on the Euclidean distance in section 5.2.1, then based on the SSIM index in section 5.3.1 and finally based on both measures combined in section 5.4.1. Section 5.4.1 will then also show the results of the probabilistic classification, the resulting disruption detection, verification and an analysis of disruption locations and causes. Finally, the most interesting differences between the analysis for week days and the analysis for weekend days will be stated in section 5.5. The full analysis for weekend days can be found in Appendix A.

5.1. Application of the three linkage methods

In this section, the three most common methods that are used to compute the cluster distances will be applied to the data set to determine whether Ward's method is indeed the most suitable. Not only a method should be chosen, the choice of a metric is also important. This metric is used to calculate the distance between two vectors. The Ward method is correctly defined only if the Euclidean pairwise metric is used (SciPy, 2019). This metric is therefore also used for the single linkage and complete linkage method to make a fair comparison.

First, the single linkage method is applied to the data of the days of the odd weeks. As explained in section 4.3, each day is represented as a vector, containing the average node, link and transfer delay for each time slice.

The results can be seen in Figure 5.1. It can be seen that the distance (on the y-axis) between data points and thus clusters appears quite small. This probably means that most of the data points will be put into one large cluster.

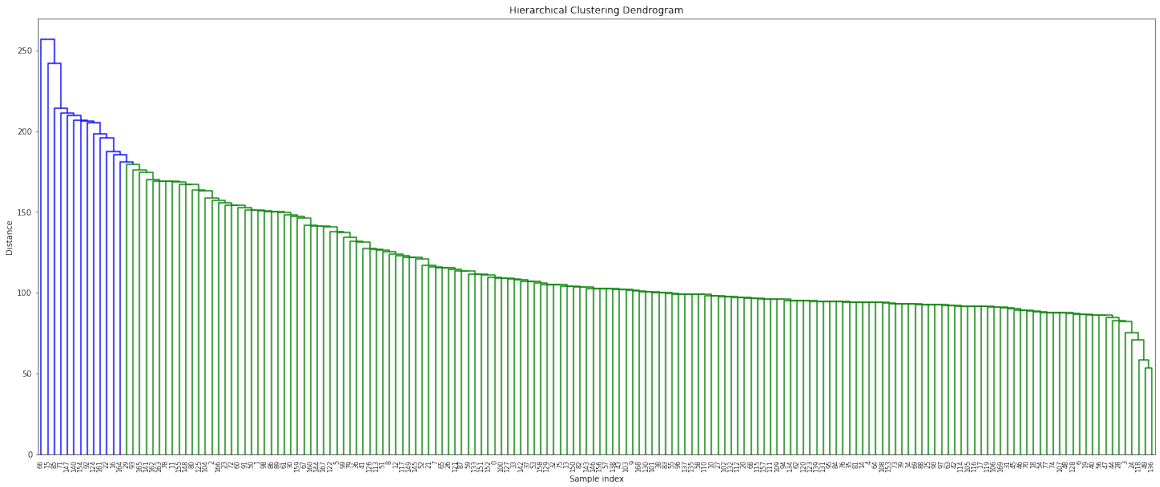


Figure 5.1: Dendrogram using the single linkage method

To create an overview of the size of the clusters, a truncated dendrogram is made, which can be found in Figure 5.2. Truncation is used to condense the dendrogram and only display the cluster sizes instead of all sample indices separately. The truncated dendrogram shows only the last p merged clusters, where p is the number of clusters, resulting from the location where the dendrogram is cut. Figure 5.2 clearly shows that using this method, almost all days are clustered together in a single cluster. Three other days are all put in a different cluster. This does not provide much information about the different types of days, so this method seems not very useful when applied to this data set.

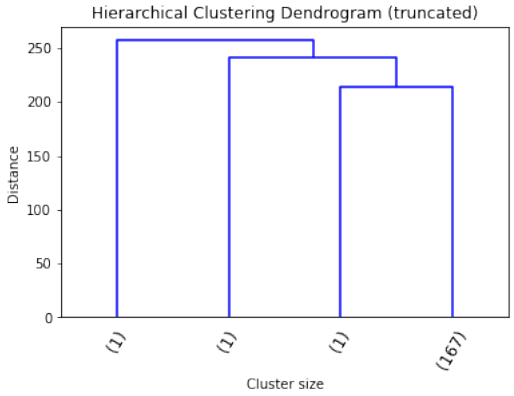


Figure 5.2: Truncated dendrogram using the single linkage method

Then, the complete linkage method is applied to the same data. The results are shown in Figure 5.3. This figure shows relatively larger distances between data points than when using the single linkage method. The largest distance has a value of approximately 350. As the distances are larger between data points, it might be easier to cut this dendrogram into clusters.

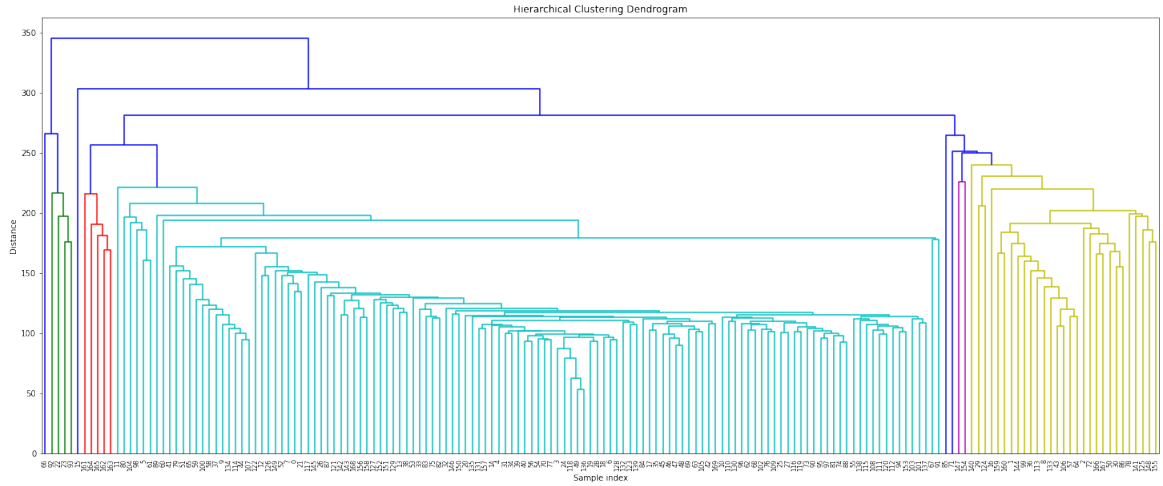


Figure 5.3: Dendrogram using the complete linkage method

The truncated dendrogram can be found in Figure 5.4. Figure 5.4 clearly shows that the complete linkage method results in two larger clusters containing almost all days. The other two clusters contain respectively five days and one single day.

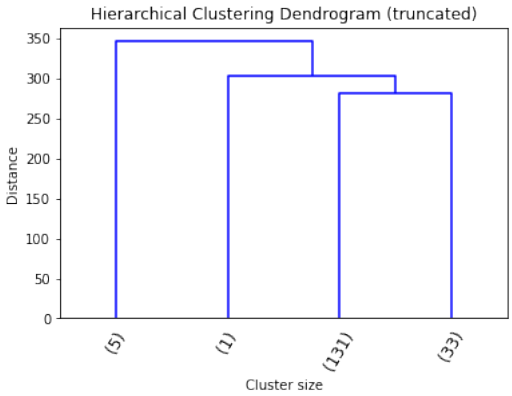


Figure 5.4: Truncated dendrogram using the complete linkage method

Finally, the Ward linkage method is applied to the data. The results can be found in Figure 5.5. This figure shows also larger distances between data points, larger than when using complete linkage, as the maximum distance value is a little higher than 600.

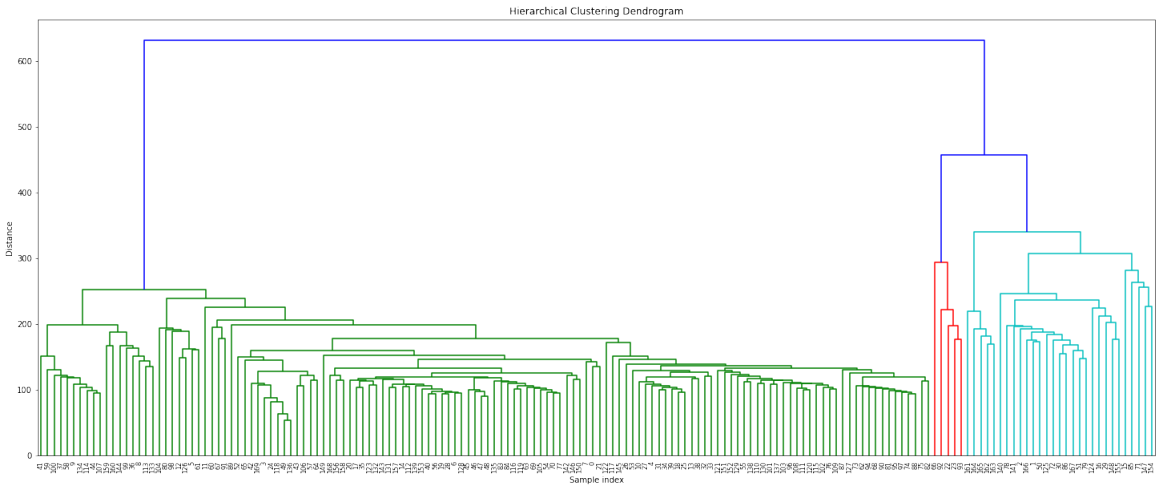


Figure 5.5: Dendrogram using the ward linkage method

The truncated dendrogram can be found in Figure 5.6. In this figure, it can be seen that there are two main clusters, one consisting of 136 days and the other of 24 days. The other two clusters both contain 5 days. The tree was cut at four clusters as adding another cluster, so creating five clusters instead of four, leads to an extra cluster that only consists of one day. It is therefore decided to create four clusters. It can be concluded that the Ward method is the most suitable to apply to this data, as this leads to relatively larger clusters when compared to the other two methods. Therefore, Ward's method is used to perform the hierarchical clustering in this research.

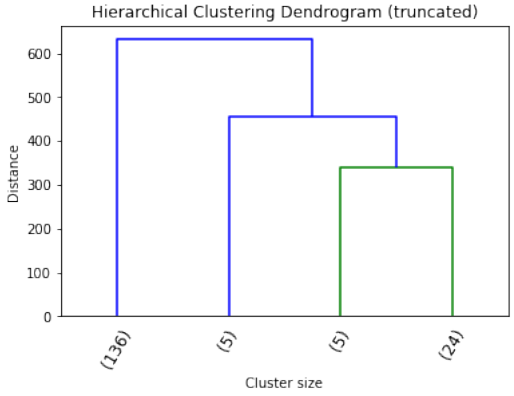
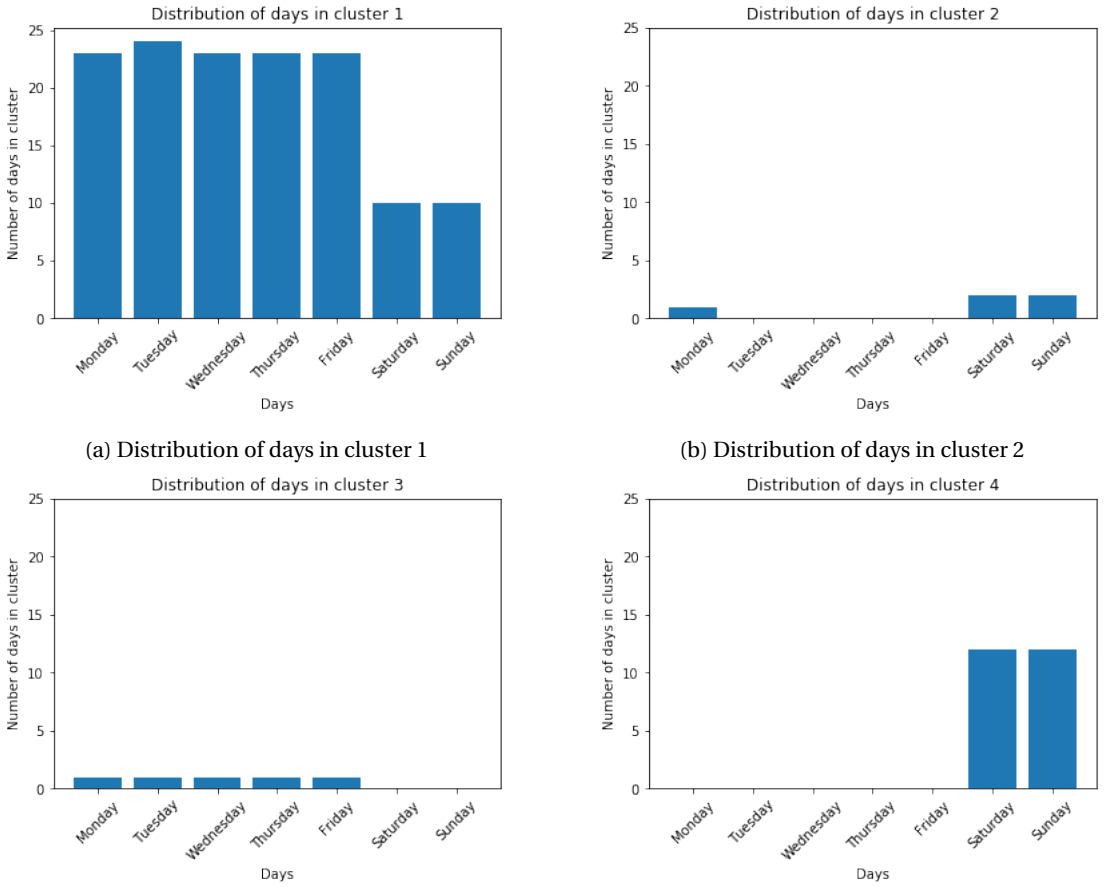


Figure 5.6: Truncated dendrogram using the ward linkage method

5.1.1. Distribution of days within clusters

Now that clusters are formed, it will be interesting to look at the distribution of days within these clusters. The results can be found in Figure 5.7. Figure 5.7a shows that cluster 1 consists of mainly weekdays and a couple of Saturdays and Sundays. Cluster 2 comprises mainly Saturdays and Sundays, while cluster 3 consists of only weekdays. Cluster 4 consists of only weekend days, equally distributed over Saturdays and Sundays.



(a) Distribution of days in cluster 1

(b) Distribution of days in cluster 2

(c) Distribution of days in cluster 3

(d) Distribution of days in cluster 4

Figure 5.7: Distribution of days in clusters

The goal of the clustering of days is to determine for a 'new' day (a day from the test set) whether it is an irregular day or not. This is done by determining if this day belongs to one of the clusters or not. When for example a Monday would fit perfectly into cluster 4, it would not be considered an irregular day, while in fact it is one, as it is not similar to an average week day but to a weekend day. To avoid this false classification, the rest of the analysis will be split up into weekdays and weekend days. In the next section, the results of the hierarchical clustering of the days in the odd weeks using Ward's method will be shown. This is done separately for both the week and weekend days.

5.2. Analysis of weekdays based on Euclidean distance

In this section, the week days will be hierarchically clustered based on the Euclidean distance. After that, the distribution of days will be shown for each cluster, and finally the centroids of the clusters will be visualised.

5.2.1. Hierarchical clustering of week days based on Euclidean distance

In Figure 5.8, the results of the hierarchical clustering, using the Euclidean distance between days, for the week days of the odd weeks are shown. It is chosen to cut the tree at four clusters, as adding another cluster would only lead to an extra cluster containing one day, resulting in two clusters that only contain one day. It can be seen that the distance between data points has been reduced when comparing to the clustering of all the days using the Ward method (the maximum distance is now approximately 400 instead of 600). This is probably due to the fact that weekend days have been removed from the data set, so the days look more similar to each other.

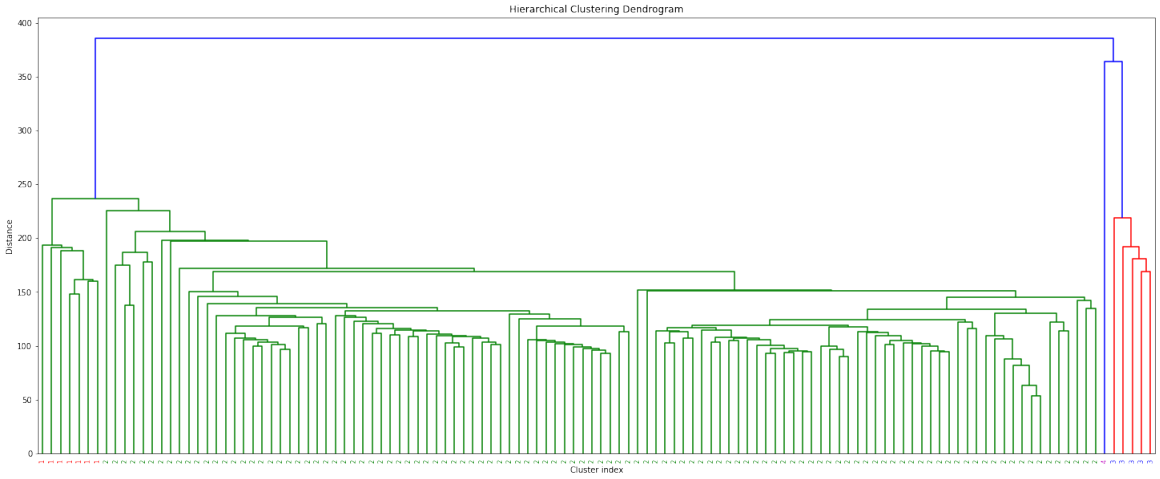


Figure 5.8: Dendrogram week days solely based on delay values

An overview of the cluster sizes can be found in the truncated dendrogram in Figure 5.9. This figure shows that there is one large cluster (cluster 2) that consists of 109 days, two smaller clusters (cluster 1 and 3) containing respectively 7 and 5 days, and one cluster (cluster 4) that consists of one day only.

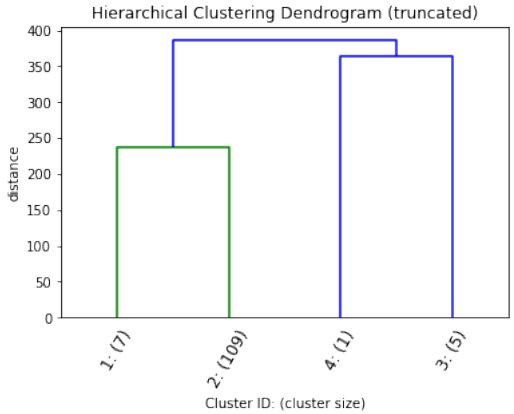
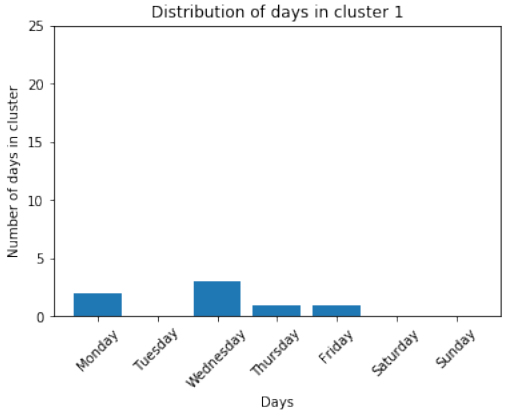


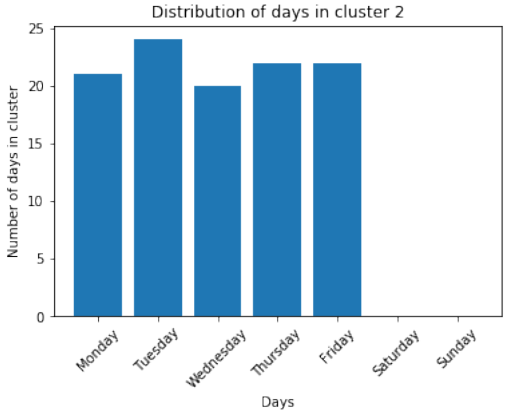
Figure 5.9: Truncated dendrogram week days solely based on delay values

5.2.2. Distribution of days for clusters of week days based on Euclidean distance

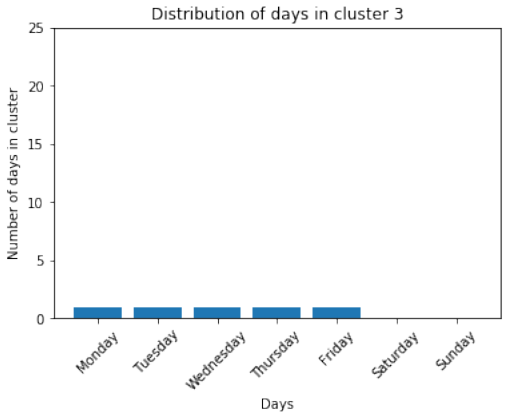
In this section, the distribution of days for each cluster, solely based on the Euclidean distance between days, will be shown. Figure 5.10 shows that cluster 1 contains three Wednesdays, two Mondays and one Thursday and one Friday. This is quite a small cluster. Cluster 2 is much larger and contains a more or less equal number of all week days. This is also the largest cluster. It can therefore be expected that the regular days can be found in this cluster. Cluster 3 contains five days, one of each week days. Cluster 4 consists of one Monday.



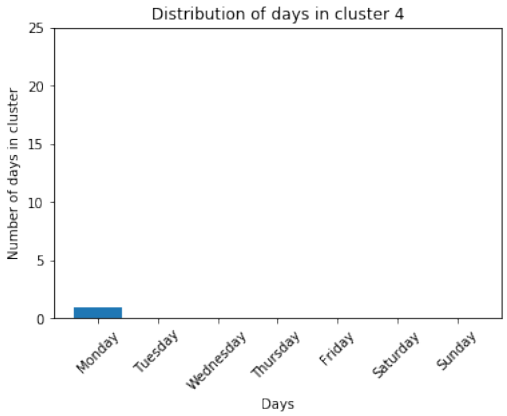
(a) Distribution of week days in cluster 1 solely based on delay values



(b) Distribution of week days in cluster 2 solely based on delay values



(c) Distribution of week days in cluster 3 solely based on delay values

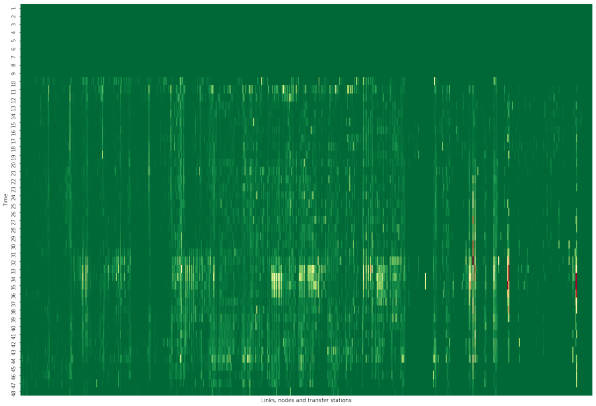


(d) Distribution of week days in cluster 4 solely based on delay values

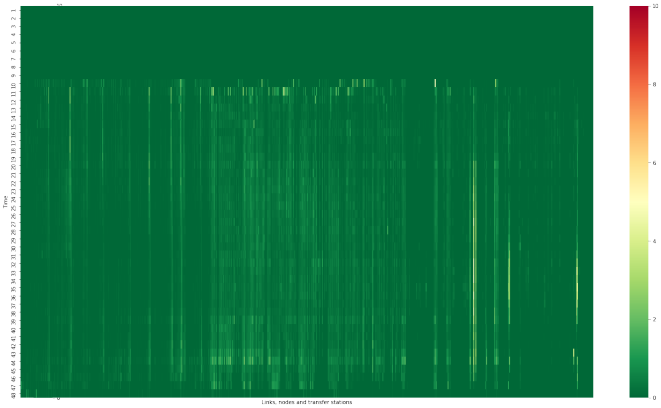
Figure 5.10: Distribution of week days in clusters solely based on delay values

5.2.3. Centroids of clusters based on Euclidean distance for week days

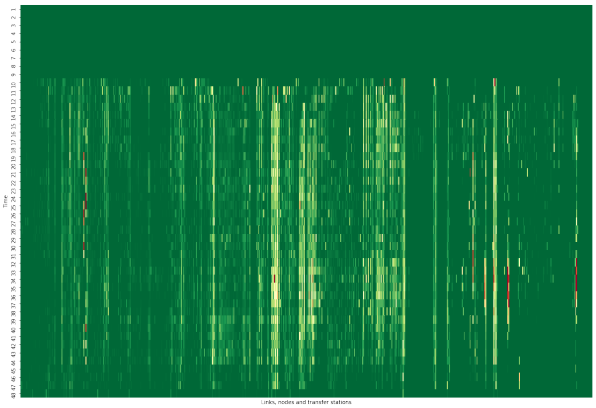
In this section, the centroids of clusters solely based on Euclidean distance will be shown. The centroids show the time slices on the y-axis and the links, nodes and transfer stations on the x-axis. The number of links, nodes and transfer stations is equal to 193, so the total length of the x-axis is 579. The first 193 points represent the 193 links, the second 193 points the 193 nodes (stations) and the last 193 points are the transfer stations. These links, nodes and transfer stations are numbered from 1 to 193 and are not in order of the links, nodes and transfer stations in the network, so they are not necessarily connected in that order. This means that the centroid does not tell anything about the spatial distribution of the delays, but only shows the temporal distribution. The spatial distribution will be shown by creating graphs per time slice of the delays in the network, for the centroids of clusters resulting from the combined clustering. Figure 5.11d shows many delays of 10 minutes and more. This image represents cluster 4 that consists of one day only, so this day is probably an irregular day. Furthermore, it can be seen that cluster 1 has a large number of node delays around 5 minutes in the evening peak, between 4 and 6 pm. During this period, there are also a few large transfer delays of around 10 minutes. Cluster 3 has many delays overall, especially a lot of node delays, of approximately 5 minutes. Cluster 2 has the smallest delays. This is also the largest cluster by far, so this cluster most likely represents regular days.



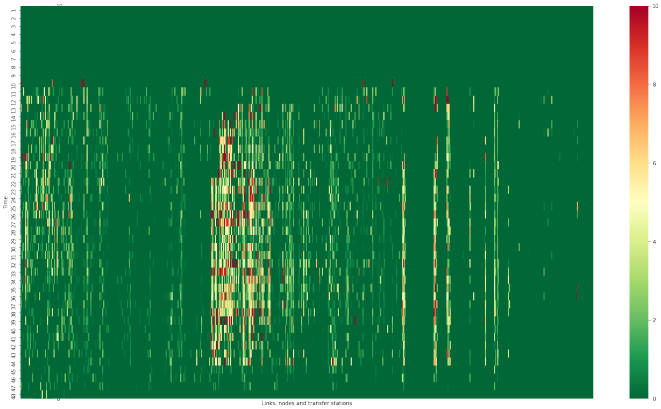
(a) Centroid of cluster 1 solely based on delay values



(b) Centroid of cluster 2 solely based on delay values



(c) Centroid of cluster 3 solely based on delay values



(d) Centroid of cluster 4 solely based on delay values

Figure 5.11: Centroids of clusters of week days solely based on delay values

5.3. Analysis of weekdays based on SSIM index

In this section, the week days will be hierarchically clustered based on the SSIM index. After that, the distribution of days will be shown for each cluster, and finally the centroids of the clusters will be visualised.

5.3.1. Hierarchical clustering of week days based on SSIM index

In Figure 5.12, the results of the hierarchical clustering for the week days of the odd weeks are shown, based on the SSIM index. It is chosen to cut the tree at five clusters, as this is an understandable number of clusters and it might result in distinct weekdays with clusters that contain for example only Mondays or only Tuesdays etcetera. This figure shows a much smaller maximum distance between data points, which makes sense as the SSIM index has a maximum value of 1.

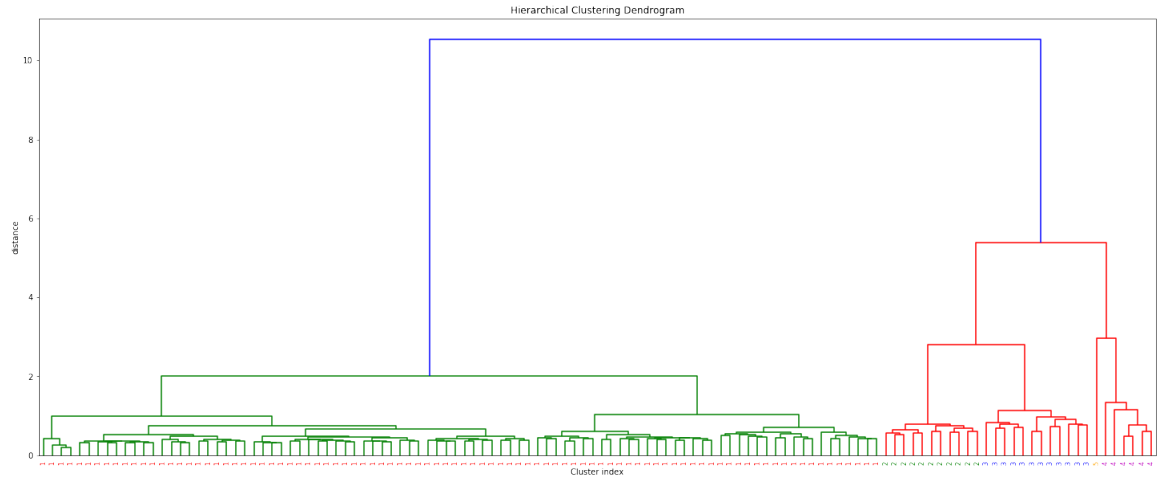


Figure 5.12: Dendrogram week days based on SSIM index

An overview of the cluster sizes can be found in the truncated dendrogram in Figure 5.13. This figure shows that there is one large cluster (cluster 1) which consists of 92 days, two smaller clusters that contain 11 days (cluster 2) and 12 days (cluster 3), an even smaller cluster containing 6 days (cluster 4) and one cluster (cluster 5) that consists of only one day.

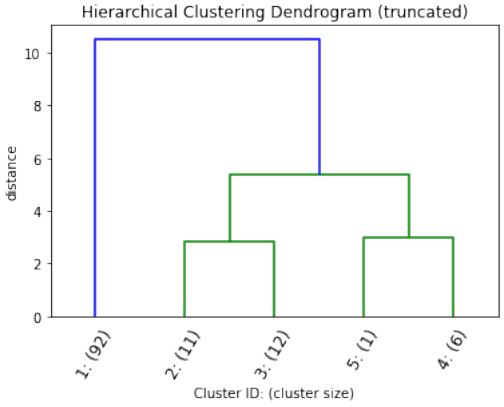
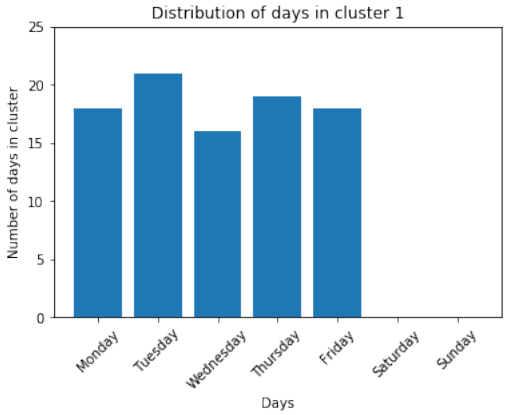


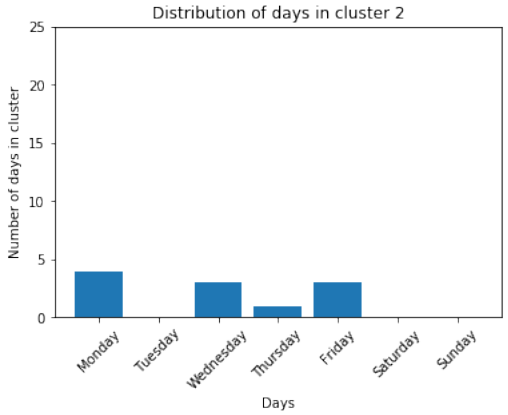
Figure 5.13: Truncated dendrogram week days based on SSIM index

5.3.2. Distribution of days for clusters of week days based on SSIM index

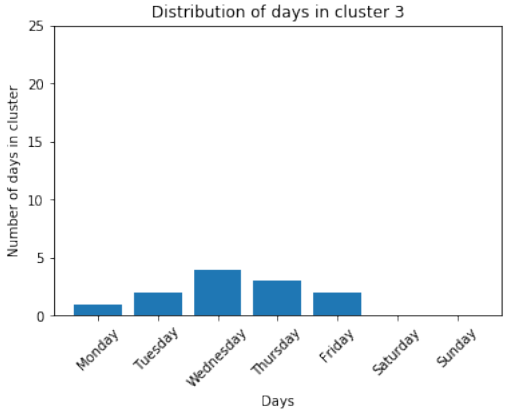
In this section, the distribution of days for each cluster, based on the SSIM index, will be shown. Figure 5.14 shows that the weekdays are more or less equally distributed over cluster 1. Cluster 2 contains 4 Mondays, 3 Wednesdays and 3 Fridays and one Thursday. Cluster 3 contains 4 Wednesdays, 3 Thursdays, 2 Tuesdays and 2 Fridays and one Monday. Cluster 4 consists of two Tuesdays and one of each of the other days. Cluster 5 contains one Monday only.



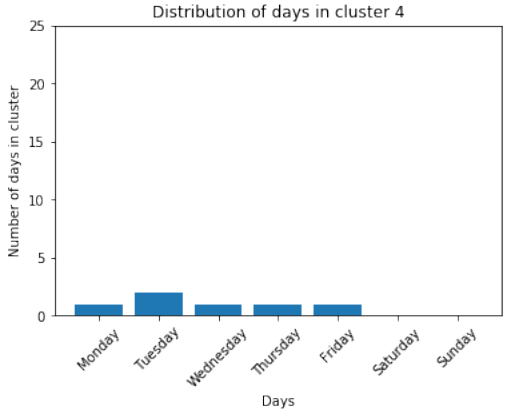
(a) Distribution of week days in cluster 1 based on SSIM index



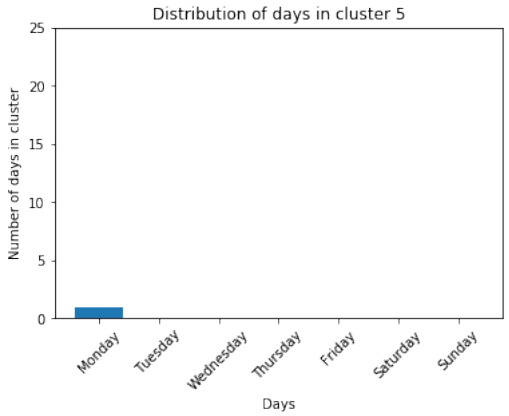
(b) Distribution of week days in cluster 2 based on SSIM index



(c) Distribution of week days in cluster 3 based on SSIM index



(d) Distribution of week days in cluster 4 based on SSIM index

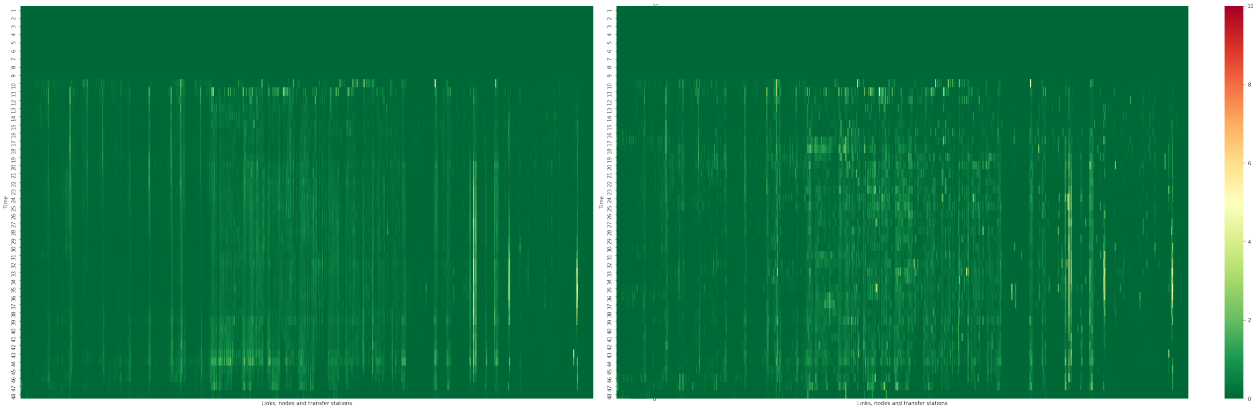


(e) Distribution of week days in cluster 5 based on SSIM index

Figure 5.14: Distribution of week days in clusters based on SSIM index

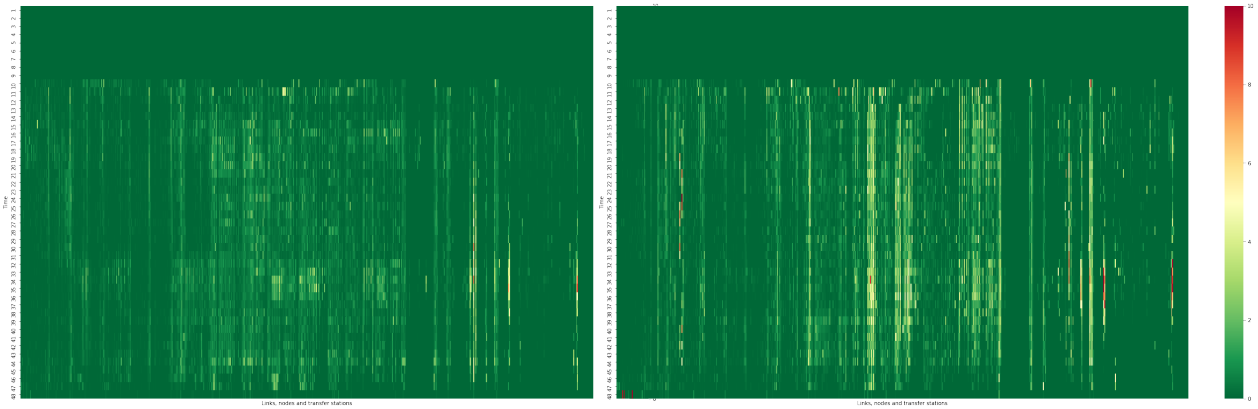
5.3.3. Centroids of clusters based on SSIM index for week days

In this section, the centroids of week day clusters based on the SSIM index will be shown. Figure 5.15 shows that cluster 5 contains the largest delays. This is also the smallest cluster, consisting of one day only. Cluster 2 and 3 look quite similar. Cluster 4 has many delays around 5 minutes. Cluster 1 has the smallest delays, this cluster most likely represents the regular days.



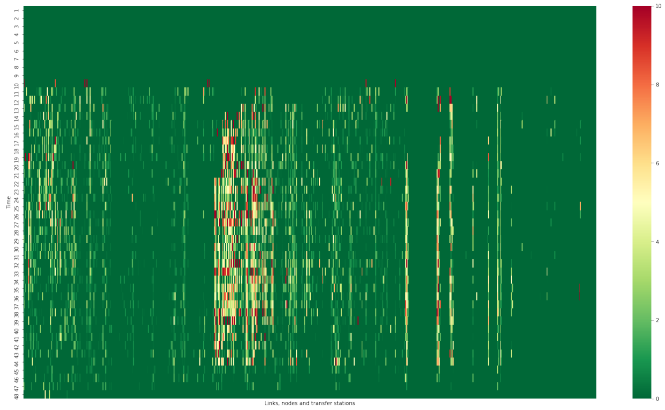
(a) Centroid of cluster 1 based on SSIM index

(b) Centroid of cluster 2 based on SSIM index



(c) Centroid of cluster 3 based on SSIM index

(d) Centroid of cluster 4 based on SSIM index



(e) Centroid of cluster 5 based on SSIM index

Figure 5.15: Centroids of clusters of week days based on SSIM index for week days

5.4. Analysis of weekdays based on Euclidean distance and SSIM index

In this section, the week days will be hierarchically clustered based on the Euclidean distance and SSIM index, so the combined clustering will be applied. After that, the distribution of days will be shown for each cluster, and the centroids of the clusters will be visualised. After that, histograms will be shown for the stacked link, node and transfer delays for each week day cluster. Then, graphs of the network will be shown to indicate locations where delays occur at certain time periods. Then, probabilistic classification will be applied to find irregular week days. This is followed by detecting disruptions within those irregular days, after which these disruptions are connected in both a spatial as well as a temporal way. Finally, the detected disruptions will be verified. The same analysis for weekend days can be found in Appendix A.

5.4.1. Hierarchical clustering of week days based on both Euclidean distances and SSIM index

In Figure 5.16, the results of the hierarchical clustering for the week days of the odd weeks are shown, based on both the Euclidean distance and the SSIM index. This figure shows that the values of some of the data points are exactly the same, as the distance between those data points is zero. This can be explained, as the data points are reduced to a vector with a length of 2 (instead of 579) with respectively 4 and 5 different labels (cluster IDs). So the maximum number of different data points is 20. It is chosen to cut the tree at five clusters, as adding another cluster would split up cluster 2 into two clusters containing 1 and 6 days. A larger cluster of 7 days is preferred over two clusters of 1 and 6 days as a cluster of 1 day gives less information about that cluster.

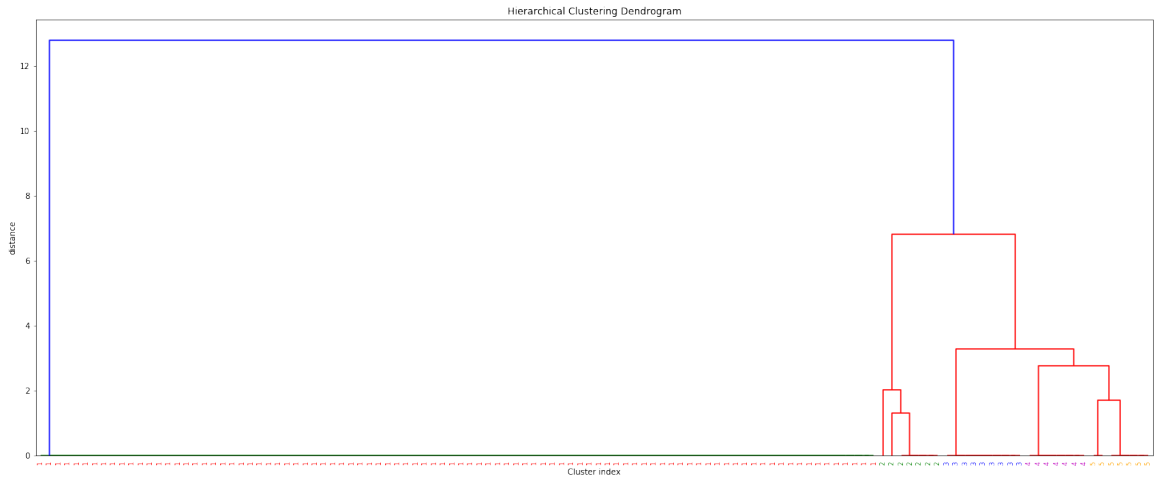


Figure 5.16: Dendrogram week days based on both Euclidean distance and SSIM index

An overview of the cluster sizes can be found in the truncated dendrogram in Figure 5.17. This figure shows that there is one larger cluster, cluster 1, containing 92 days. The other 4 clusters are quite similar in size, three clusters contain each 7 days (cluster 2, 4 and 5) and one (cluster 3) contains 9 days.

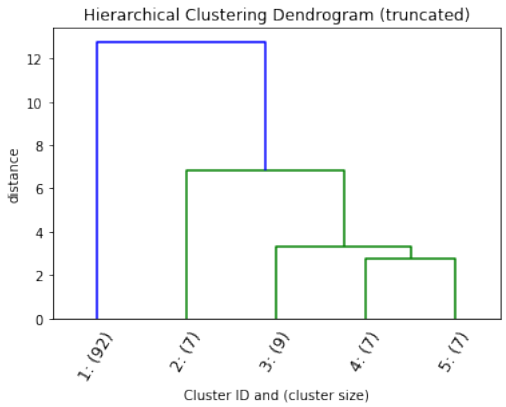
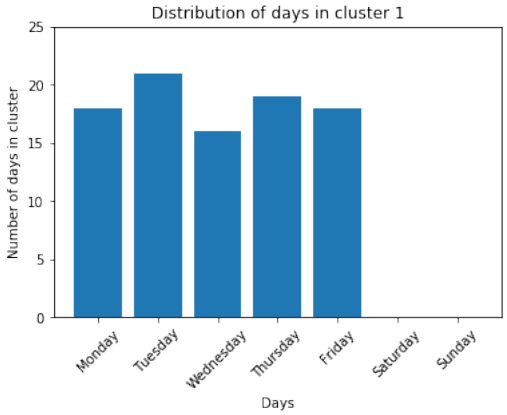


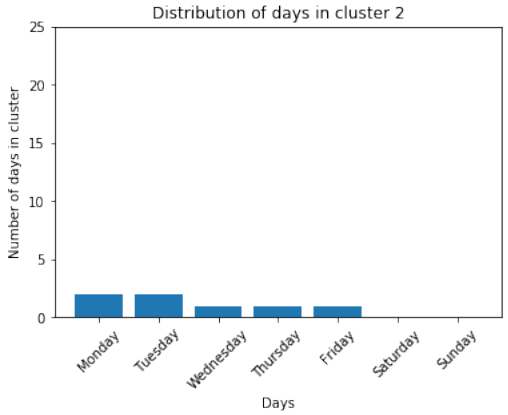
Figure 5.17: Truncated dendrogram week days based on both Euclidean distance and SSIM index

5.4.2. Distribution of days for clusters of week days based on Euclidean distance and SSIM index

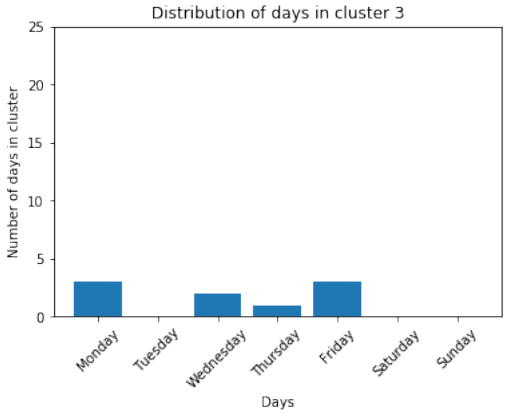
In this section, the distribution of days for each cluster, based on both the Euclidean distance and the SSIM index, will be shown.



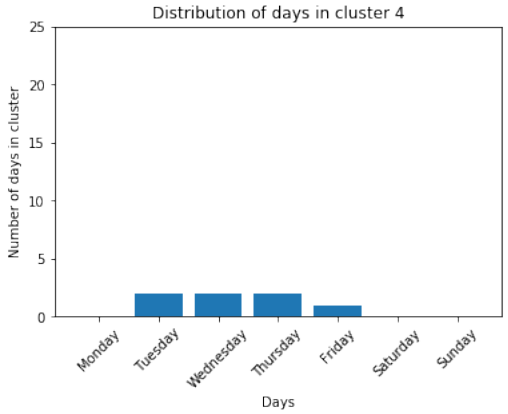
(a) Distribution of week days in cluster 1 based on Euclidean distance and SSIM index



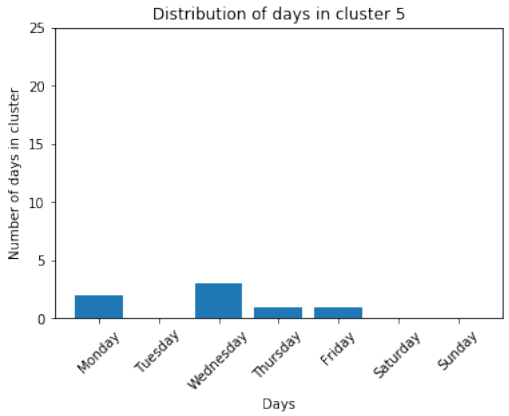
(b) Distribution of week days in cluster 2 based on Euclidean distance and SSIM index



(c) Distribution of week days in cluster 3 based on Euclidean distance and SSIM index



(d) Distribution of week days in cluster 4 based on Euclidean distance and SSIM index



(e) Distribution of week days in cluster 5 based on Euclidean distance and SSIM index

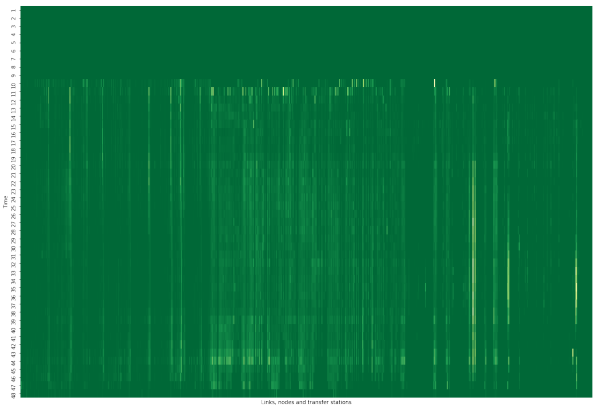
Figure 5.18: Distribution of week days in clusters based on Euclidean distance and SSIM index

Figure 5.18 shows that in cluster 1, the days are quite equally distributed. Cluster 2 contains 2 Mondays, 2 Tuesdays and one of each other day. Cluster 3 consists of 3 Mondays and 3 Fridays, 2 Wednesdays and one

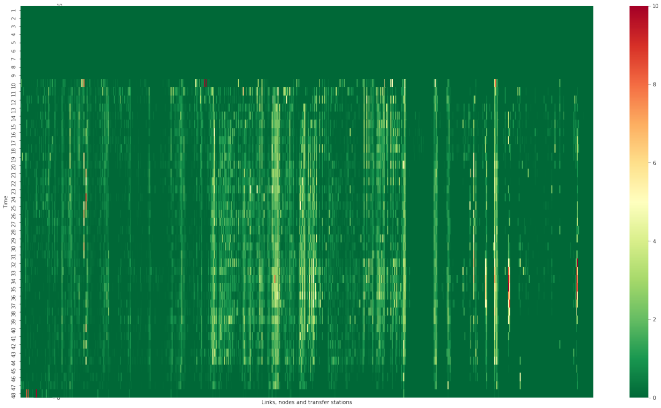
Thursday. Cluster 4 consists of 2 Tusedays, 2 Wednesdays and 2 Thursdays and one Friday. Cluster 5 contains 2 Mondays, 3 Wednesdays, one Thursday and one Friday.

5.4.3. Centroids of clusters based on Euclidean distance and SSIM index for week days

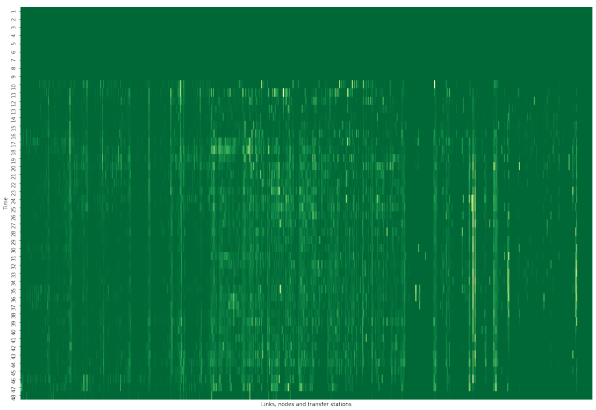
In this section, the centroids of clusters based on the Euclidean distance and SSIM index will be shown. Figure 5.19 shows that the centroid of cluster 2 contains the largest delays, followed by cluster 5. Cluster 3 and 4 contain less delays, while cluster 1 contains the least delays. To get a better insight in which kind of delays (node, link or transfer delays) occur at what time slices, a histogram of the stacked link, node and transfer delays for each time slice is created for each cluster.



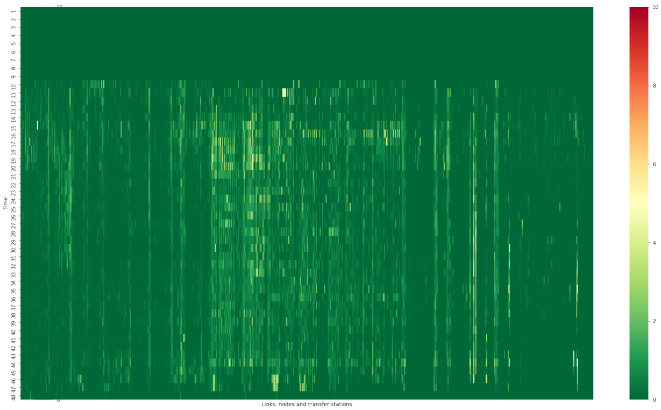
(a) Centroid of cluster 1 based on Euclidean distance and SSIM index



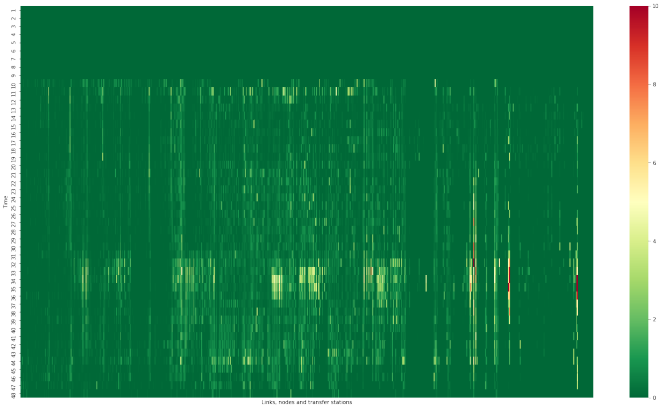
(b) Centroid of cluster 2 based on Euclidean distance and SSIM index



(c) Centroid of cluster 3 based on Euclidean distance and SSIM index



(d) Centroid of cluster 4 based on Euclidean distance and SSIM index



(e) Centroid of cluster 5 based on Euclidean distance and SSIM index

Figure 5.19: Centroids of clusters of week days based on Euclidean distance and SSIM index

5.4.4. Histograms of the stacked link, node and transfer delays for each week day cluster

In this section, the histograms of the stacked link, node and transfer delays are shown for each cluster. Figures 5.20,5.21,5.22,5.23 and 5.24, show that overall, the node delays are the largest of all three delays. Furthermore, it can be seen that cluster 1 has the largest delays in the morning between 5 am and 5:30 am and in the evening between 9:30 pm and 10 pm. Cluster 2 has large delays overall, but mainly in the afternoon/evening peak (between 3:30 pm and 7:30 pm). Cluster 3 looks a bit similar to cluster 4, but has much smaller delays than cluster 4. Cluster 3 shows a peak between 5:00 am and 5:30 am and again between 8:00 am and 10 am. A small afternoon peak can be distinguished between 4:30 pm and 7:30 pm. Cluster 4 shows a morning peak between 7 am and 10 am and a peak in the evening between 9:30 pm and 10 pm. Cluster 5 shows a large afternoon peak between 3:30 pm and 6 pm.

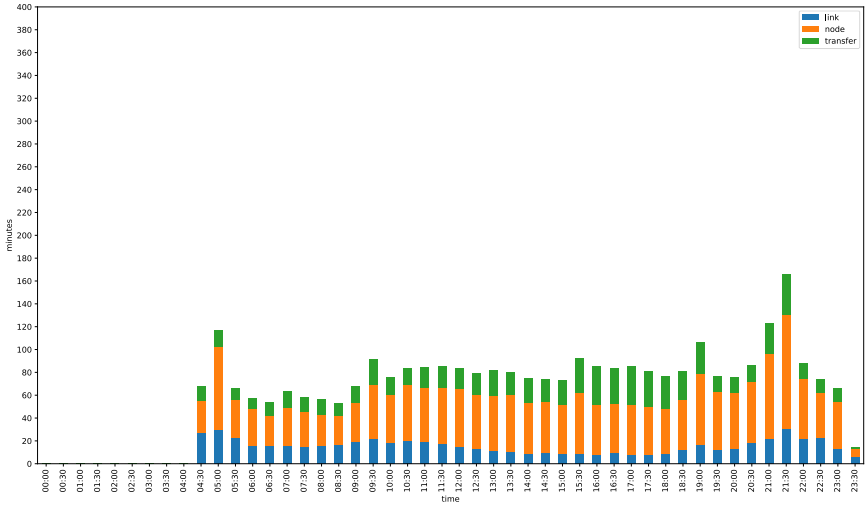


Figure 5.20: Stacked histogram for link, node and transfer delays per time slice of cluster 1

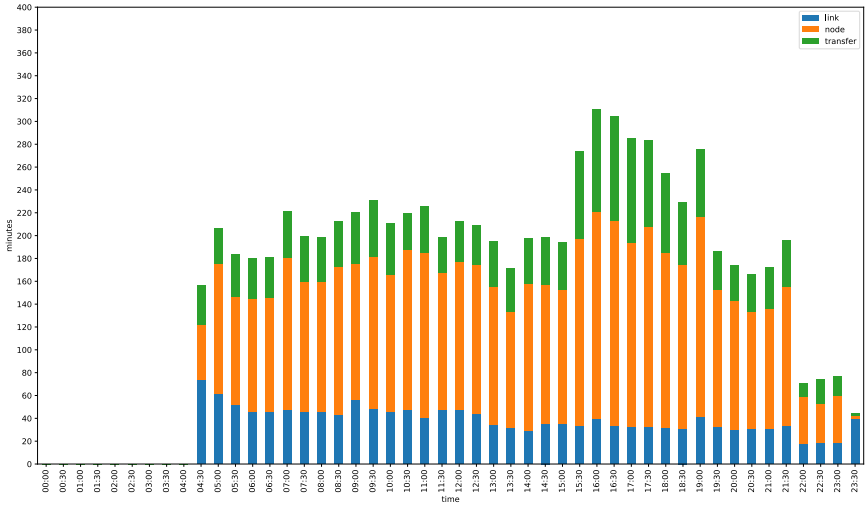


Figure 5.21: Stacked histogram for link, node and transfer delays per time slice of cluster 2

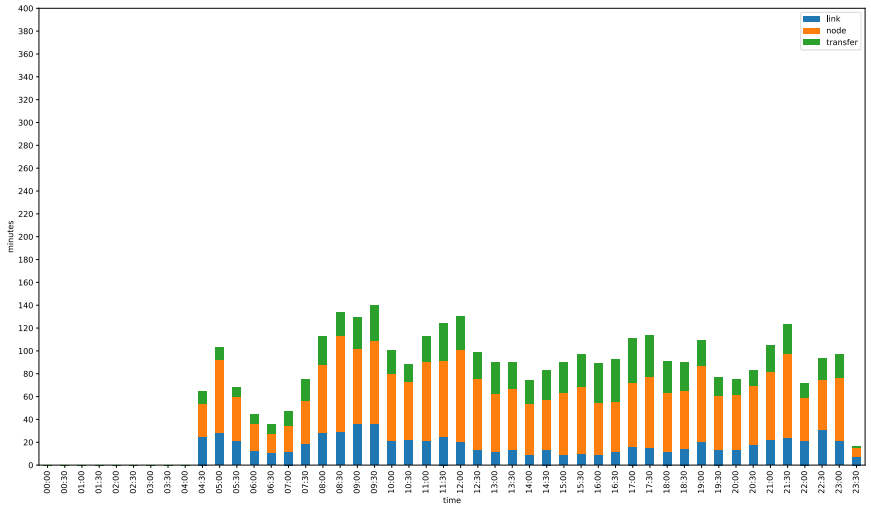


Figure 5.22: Stacked histogram for link, node and transfer delays per time slice of cluster 3

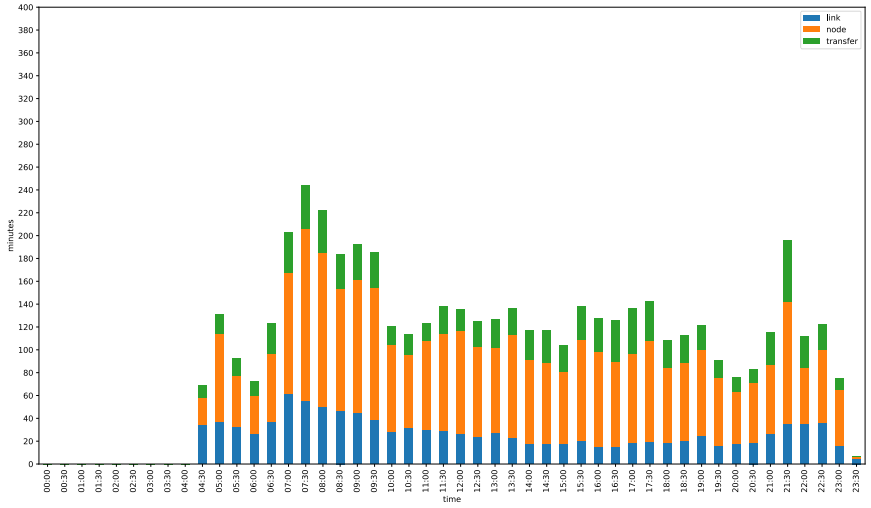


Figure 5.23: Stacked histogram for link, node and transfer delays per time slice of cluster 4

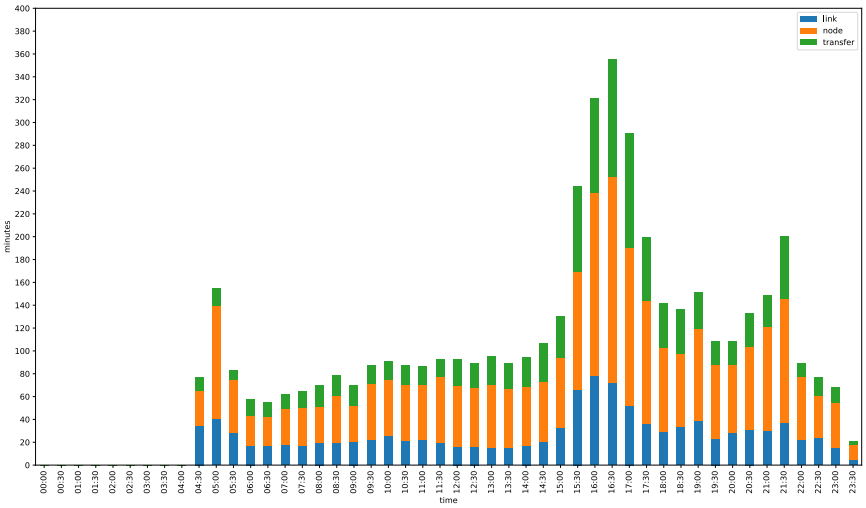
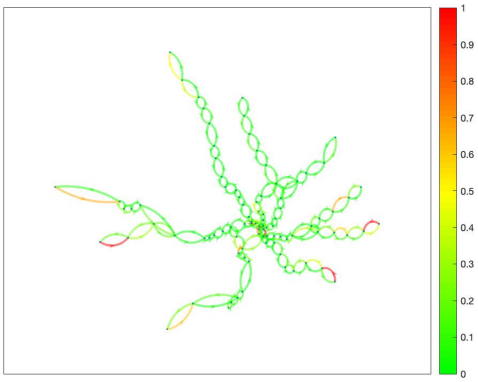


Figure 5.24: Stacked histogram for link, node and transfer delays per time slice of cluster 5

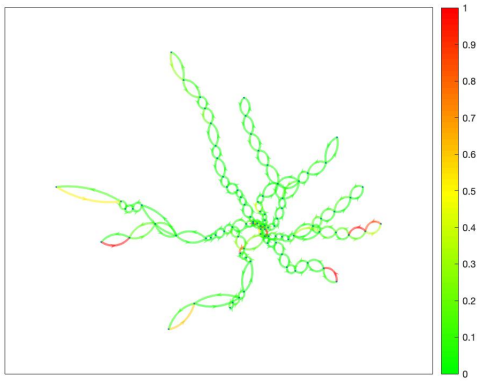
After determining the time slices that contain the largest delays, it will be interesting to look at the locations where these delays occur. In the next section, graphs will be shown of certain time slices to indicate these locations.

5.4.5. Graphs of the network indicating locations where delay occurs at certain time slices

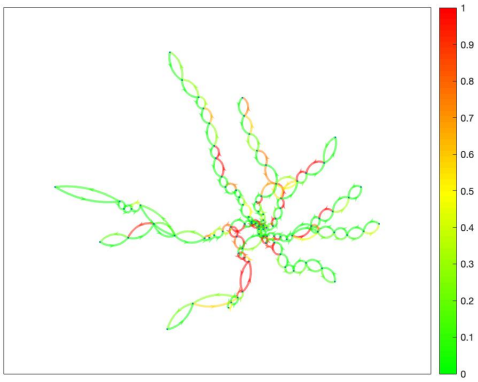
In this section, graphs will be shown that indicate the locations where certain types of delay occur for each cluster.



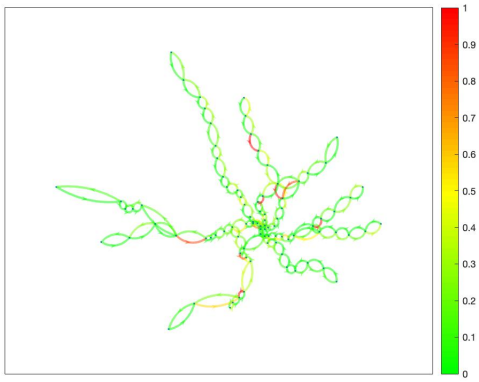
(a) Network graph representing link delays between 5 am and 5:30 am for cluster 1



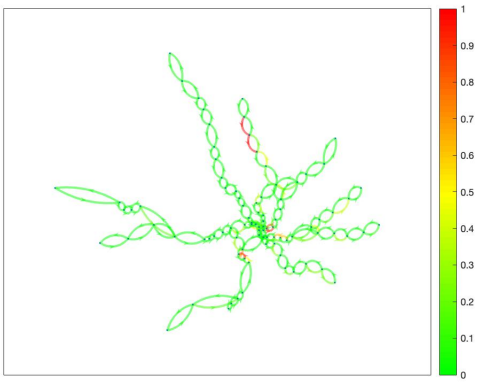
(b) Network graph representing link delays between 5:30 am and 6 am for cluster 1



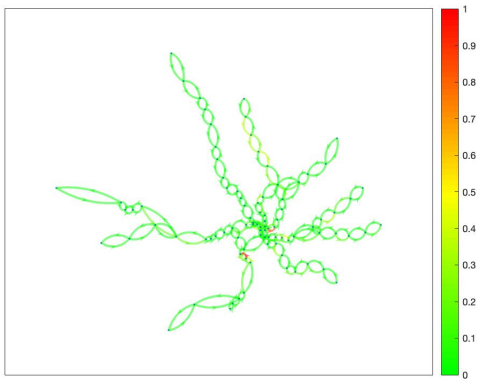
(c) Network graph representing node delays between 5 am and 5:30 am for cluster 1



(d) Network graph representing node delays between 5:30 am and 6 am for cluster 1



(e) Network graph representing transfer delays between 5 am and 5:30 am for cluster 1



(f) Network graph representing transfer delays between 5:30 am and 6 am for cluster 1

Figure 5.25: Network graphs representing link, node and transfer delays between 5 am and 6 am for cluster 1

In cluster 1, the largest delays occur between 5 am and 5:30 am and 9:30 pm and 10 pm. Therefore, these time slices are investigated. Separate graphs have been made for each type of delay, the node, link and transfer delays.

Figure 5.25 shows that link delays between 5 and 5:30 am mostly occur on the outskirts of the network, at stations in the East, a station in the South East and stations in the West. It can be seen that in the next time slice, mainly the link delay at the station in the East spreads to its adjacent station. The node delays between 5 am and 5:30 are widely spread over the network. In the next time slice, it can be seen that the node delays propagate more to the centre of the graph. Furthermore, it can be seen that the largest transfer delays between 5 and 5:30 am can be found at stations in the North. In the adjacent time slice, the transfer delays have been largely reduced. These graphs have been made for all clusters and all time slices with peak delays. These can be found in Appendix B. A summary of the findings of these graphs can be found in Tables 5.1, 5.2,5.3,5.4 and 5.5.

Delay peak time	**Locations link delay**	**Locations node delay**	**Locations transfer delay**
5 - 5:30 am	West, South, East	Spread, mainly centre	North
9:30 - 10 am	West, South, East	Spread, mainly centre	North, West

Table 5.1: Delay peak times and locations cluster 1 for week days

Delay peak time	**Locations link delay**	**Locations node delay**	**Locations transfer delay**
5:30 - 7:30 pm	North, West, East, Centre	North, West, East, Centre	North, West, East, Centre

Table 5.2: Delay peak times and locations cluster 2 for week days

Delay peak time	**Locations link delay**	**Locations node delay**	**Locations transfer delay**
5 - 5:30 am	West, South, East	North, South, Centre	North
8 - 10 am	North, East	North, Centre	North, West, Centre
9:30 - 10 pm	West, South, East	North, West, East, Centre	North, West, East, Centre

Table 5.3: Delay peak times and locations cluster 3 for week days

Delay peak time	**Locations link delay**	**Locations node delay**	**Locations transfer delay**
7 - 10 am	West, South, East	North, West, Centre	North, West, Centre
9:30 - 10 pm	West, South East	North, West, East, Centre	North, West, East, Centre

Table 5.4: Delay peak times and locations cluster 4 for week days

Delay peak time	**Locations link delay**	**Locations node delay**	**Locations transfer delay**
3:30 - 6 pm	West, South, East	Whole network	West, Centre, East

Table 5.5: Delay peak times and locations cluster 5 for week days

Table 5.6 shows a summary for each cluster, including the number of days within each cluster, the total link, node and transfer delay, the total delay of these three delays and the peak periods within the clusters.

	Cluster 1	**Cluster 2**	**Cluster 3**	**Cluster 4**	**Cluster 5**
Number of days	92	7	9	7	7
Total link delay (min)	607	1533	720	1096	1098
Total node delay (min)	1747	4584	2010	2925	2476
Total transfer delay (min)	737	1684	845	944	1136
Total delay (min)	3091	7801	3574	4966	4711
Peak period(s)	Morning and evening	Evening	Morning and evening	Morning and evening	Evening

Table 5.6: Summary of each cluster for week days

5.4.6. Probabilistic classification of week days

The centroids of each cluster based on both delay values and the SSIM index are used to detect disruptions. With probabilistic classification, the probabilities of belonging to each of the clusters are calculated for each day. To determine whether a day that has a probability to belong to a cluster is irregular or not, it must be determined whether the cluster represents irregular days or regular days. Cluster 2, 3, 4 and 5 are all much smaller clusters than cluster 1 and contain larger delays. Therefore, it is assumed that cluster 1 represents regular days and cluster 2, 3, 4 and 5 represent irregular days. When a day is classified as an irregular day when the probability of belonging to cluster 1 is between 30% and 70%, this leads to 9 irregular days (of the 124 days that are analysed). This means that with this data, it makes no difference if a threshold of 30% or 70% is chosen, both lead to the same number of irregular days. The dates of these days including a possible explanation why this day is irregular can be found in Table 5.7. Note that when a box is filled with '-', this does not necessarily mean that there was no track work, but that this is not known.

Date	**Holiday**	**Track work**
04-09-17	Labor day	-
25-12-17	Christmas	-
19-02-18	President's day	-
09-03-18	-	-
20-08-18	-	Yes
21-08-18	-	Yes
22-08-18	-	Yes
23-08-18	-	Yes
24-08-18	-	Yes

Table 5.7: Irregular week dates and possible explanation why these dates are irregular

It is remarkable that 3 of the detected irregular days are holidays. When applying the probabilistic classification that is created using weekend days, it is found that Labor day has a probability of 74% of belonging to cluster 1 of the weekend days, a regular weekend day cluster. Christmas day has the largest probability to belong to weekend day cluster 2 (31%). This probability is lower than 50% and on top of that, cluster 2 is an irregular weekend day cluster. So Christmas day would also have been classified as an irregular day by the weekend day algorithm. President's day has a probability of 97.5% to belong to cluster 3, a regular weekend day cluster. To sum up, regarding the holidays, only Christmas day would have been classified as an irregular day when the weekend day algorithm would have been applied. Now that the irregular days have been determined, the disruptions need to be found within these days. Therefore, the centroid from the regular cluster with the highest probability is subtracted from the delay matrix of the irregular day. Because a centroid from an irregular cluster does not contain regular delays, only centroids of regular clusters with the highest probability are being compared. As only cluster 1 represents regular days, all irregular days are compared to the centroid of this cluster. This comparison results in a matrix that consists of disruptions when compared to an average day of that class (the centroid).

5.4.7. Disruption detection in week days

To show the link, node and transfer delay of the resulting matrix consisting of disruptions when compared to the average of the cluster it belongs to, a stacked histogram is made. This histogram from the first irregular day, 4^{th} of September, 2017 is shown in Figure 5.26. This day has the highest probability of belonging to cluster 3, but this is an irregular cluster. Therefore this day is compared to the centroid of cluster 1. It can be seen that before 7:30 am, all delays are smaller than the delays in centroid 1. However, after that much larger delays occur until 9 pm.

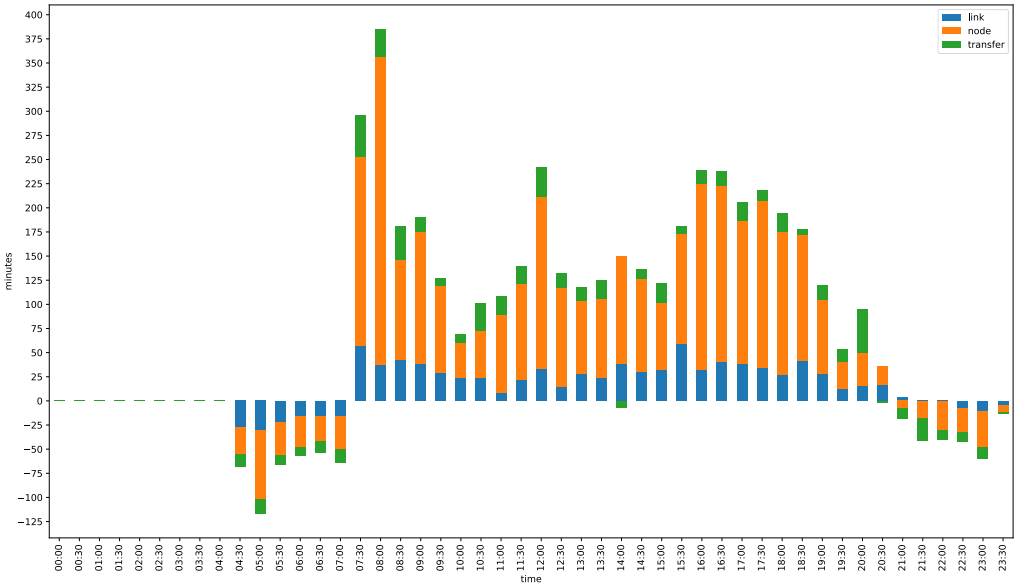
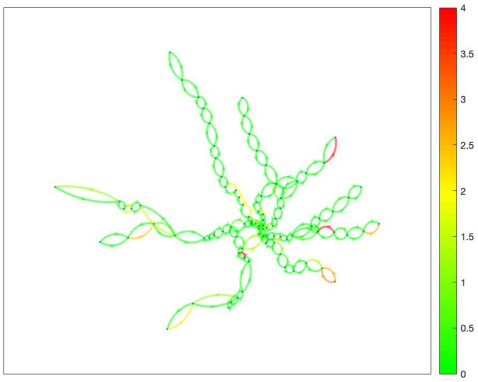
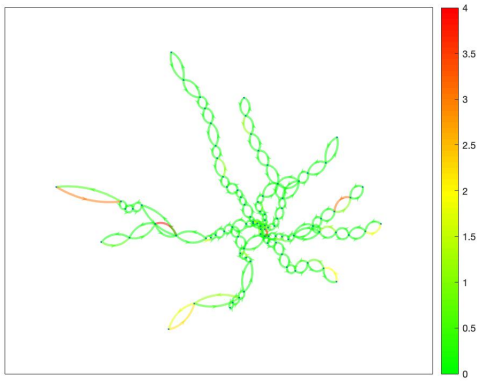


Figure 5.26: Stacked histogram of link, node and transfer delays of 04-09-17

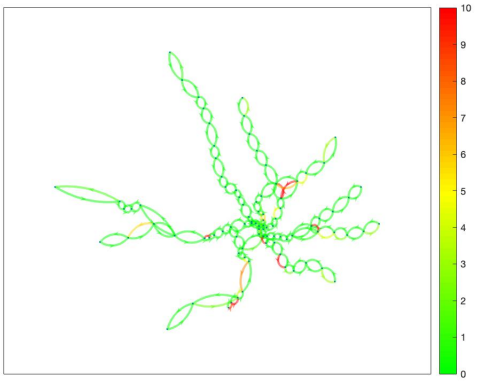
To see where delays occurred, network graphs are made for some time slices. It can be seen that the largest delays occur between 7:30 and 8:30. Therefore, these time slices are investigated. The resulting graphs can be found in Figure 5.27. The colour bar indicates the average delay in minutes. Note that the maximum value of the colour bar has been chosen relatively to the delay sizes. The colour bars do not differ per time slice, but they differ per type of delay (link, node or transfer delay). It can be seen that the largest link delays between 7:30 am and 8 am occur at a station in the North East and at a station in the East. Between 8 am and 8:30 am these delays have been largely reduced. However, new disruptions occur at stations in the West and a station in the East. Furthermore, the largest node delays between 7:30 am and 8 am occur at a station in the East and a station in the North East, and stations in the South. Between 8 am and 8:30 am, the node delay at the North East might have propagated to an adjacent station, as one of the largest node delays can be observed at this adjacent station. Furthermore, some new delays have occurred in the centre of the network. The largest transfer delays between 7:30 am and 8 am occur at stations in the North, a station in the West and stations in the centre.



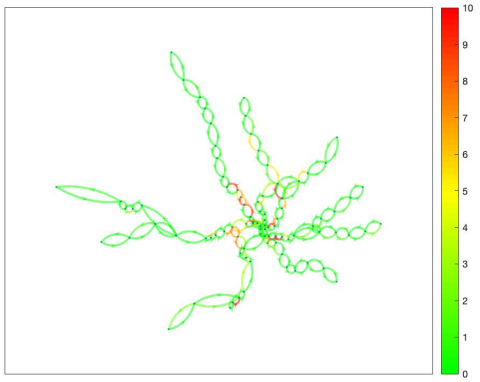
(a) Network graph representing link delays between 7:30 am and 8 am for 04-09-17



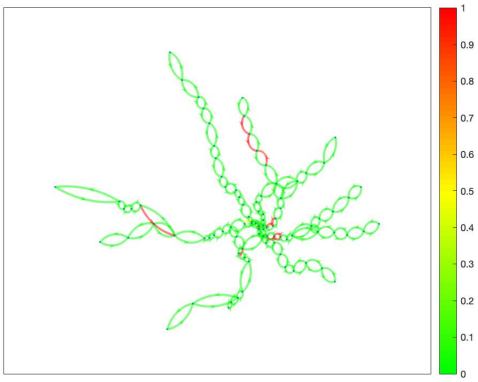
(b) Network graph representing link delays between 8 am and 8:30 am for 04-09-17



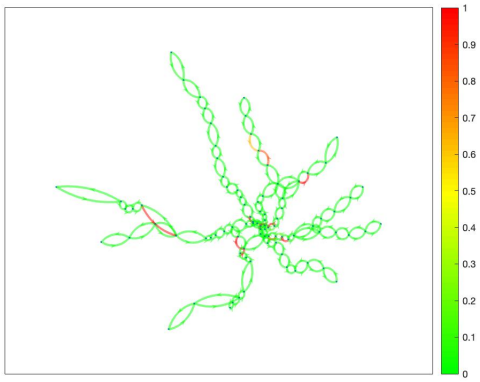
(c) Network graph representing node delays between 7:30 am and 8 am for 04-09-17



(d) Network graph representing node delays between 8 am and 8:30 am for 04-09-17



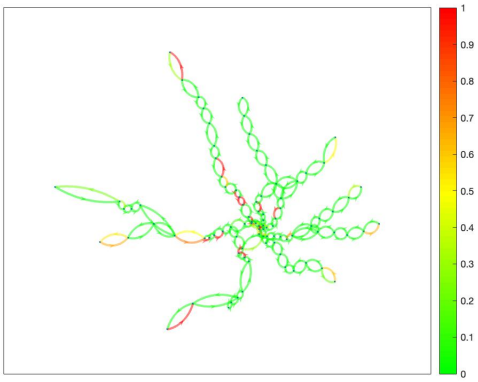
(e) Network graph representing transfer delays between 7:30 am and 8 am for 04-09-17



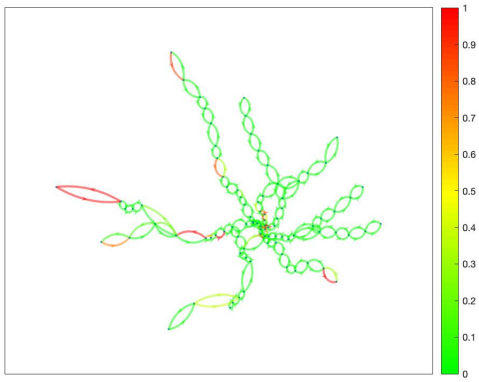
(f) Network graph representing transfer delays between 8 am and 8:30 am for 04-09-17

Figure 5.27: Network graphs representing link, node and transfer delays between 7:30 am and 8:30 am for 04-09-17

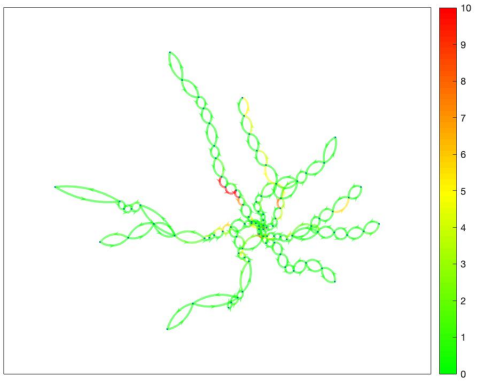
Another time slice where large delays occur, is between noon and 12:30 pm. Therefore, this time slice and the successive time slice are investigated. The resulting graphs can be found in Figure 5.28. It can be seen that the largest link delays between noon and 12:30 pm occur at stations in the North, centre and the South. In the next time slice, between 12:30 pm and 1 pm, it can be seen that the largest link delays occur at stations in the West, the North and the South East. Furthermore, the largest node delays can be seen in the northern part of the centre. According to the WMATA disruption log file, a disruption occurred at Cleveland park (in the northern part of the centre) in this time slice, caused by a propulsion system malfunction. It can be seen that in the next time slice, there are still delays in this area. When looking at the transfer delays, it can be seen that the largest transfer delays occur at a station in the West, stations in the South, North and some in the centre. In the next time slice, the transfer delay in the northern part is still there, while the transfer delay at the station in the West has been solved. A new transfer delay occurs in the centre.



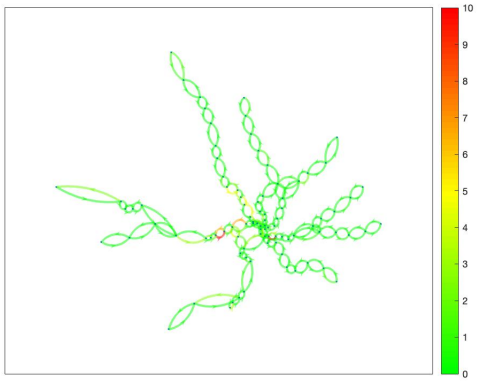
(a) Network graph representing link delays between noon and 12:30 pm for 04-09-17



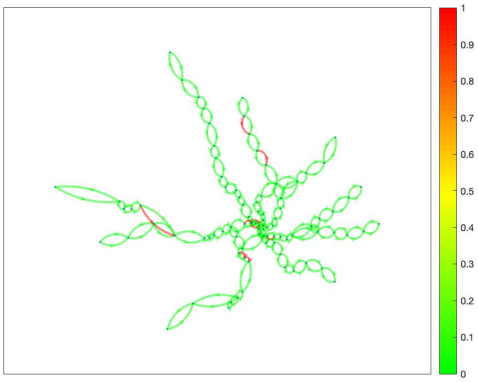
(b) Network graph representing link delays between 12:30 pm and 1 pm for 04-09-17



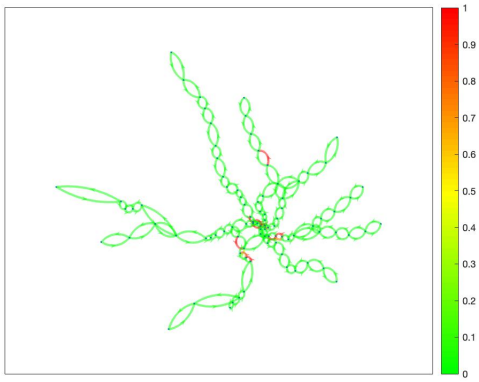
(c) Network graph representing node delays between noon and 12:30 pm for 04-09-17



(d) Network graph representing node delays between 12:30 pm and 1 pm for 04-09-17



(e) Network graph representing transfer delays between noon and 12:30 pm for 04-09-17



(f) Network graph representing transfer delays between 12:30 pm and 1 pm for 04-09-17

Figure 5.28: Network graphs representing link, node and transfer delays between noon and 1 pm for 04-09-17

5.4.8. Connecting delays in a spatial and temporal way for irregular week days

The next step is to find the locations (link, station or transfer station) that are affected by these disruptions. This is done by checking for each time slice the locations where delays occur. Thresholding is applied to reduce the sensitivity of the algorithm. This threshold is set at 0.5 minutes, as a larger threshold leads to more spatially disconnected disruptions and a lower threshold leads to a larger number of disruptions overall. After the locations are determined, algorithm 1 is applied. This results in disruptions that are spatially connected, but not yet temporally. Therefore, the next step is to apply algorithm 2, to find disruptions that are connected in both a spatial as well as a temporal way. After these both spatially and temporally connected disruptions are found, the disruption IDs, corresponding time slices, affected locations and duration are known. After that, for each location and time slice, the passenger counts at that location and time slice are multiplied with the average passenger delay at that same location and time slice, and summed for the duration of the disruption. This results in the total passenger delay and passenger counts for each disruption. The average passenger delay for each disruption can then be calculated by dividing the total passenger delay by the counts. Note that this average passenger delay is thus calculated for each network element, and not for the whole disruption itself. For a large disruption, over multiple time slices, it is not likely that one passenger will traverse all locations affected by the disruption over all time slices. The final output containing the disruptions of the 4^{th} of September, 2017 can be found in Table 5.8.

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
1	[16]	[5 8 10 12 14]	[3 6 8 10 12]	124	126	1.0	1
2	[16]	[32]	[30]	161	97	1.7	1
3	[16 17 18 19 20]	[1 2 4 5 6 7 9 10 11 12 13 14 16 18 20 29 30 31 32 34 35 37 38 40 43 44 45 46 47 49 50 51 52 53 55 56 58 60 62 63 64 66 67 68 69 70 71 72 73 74 77 79 80 82 83 84 85 86 88 89 90 91 93 94 96 98 101 102 103 104 105 109 110 112 113 115 117 119 120 121 122 126 128 129 131 132 133 134 135 136 137 138 140 142 143 144 146 148 151 153 154 155 157 158 160 164 165 166 167 168]	[1 2 3 4 5 7 8 9 10 11 12 14 16 18 27 28 29 30 32 33 34 36 39 40 41 42 43 44 45 46 47 49 50 52 54 56 58 59 60 62 63 64 65 66 67 68 69 70 73 75 76 78 79 80 81 82 84 85 86 87 89 90 94 95 96 97 98 102 103 105 106 108 110 112 113 114 115 119 121 122 124 126 127 128 129 130 131 132 133 135 137 139 140 142 143 144 146 148 151 153 154 155 157 158 160 164 165 166 167 168]	51262	94304	0.5	5
4	[16 17 18]	[168 169 171 172 173 174]	[168 169 171 172 173 174]	1095	2712	0.4	3
5	[16 17 18 19 20]	[34 35 37 38 40 43 47 52 53 54 60 63 67 68 69 70 71 72 73 74 79 80 81 82 84 85 86 88 94 96 101 102 103 108 110 112 114 117 119 120 126 128 134 135 136 137 140 148 151 152 154 155 157 158 160 162 163 164 165 167 168 169 170 172 174 176 177 178 179 180 182 183 184]	[32 33 34 36 39 43 46 47 48 54 59 63 64 65 66 67 68 69 70 75 76 77 78 80 81 82 84 90 94 95 96 101 103 105 107 110 112 113 119 121 129 130 131 132 133 137 140 148 151 152 154 155 157 158 160 162 163 164 165 167 168 169 170 172 174 176 177 178 179 180 182 183 184]	37791	58978	0.6	5

Table 5.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
6	[16 17 18 19 20]	[175 176 177 178 179 182 184 185 187 188 189]	[175 176 177 178 179 182 184 185 187 188 189]	3067	4462	0.7	5
7	[16 17 18 19 20 21 22 23 24 25 26 27 28 29 30 31]	[1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 29 30 31 32 34 35 37 38 39 40 41 42 43 44 45 46 47 48 49 52 53 54 55 56 57 58 59 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 80 81 82 83 85 86 88 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 110 111 112 113 114 115 117 118 119 120 122 124 126 127 128 129 130 131 132 133 134 135 136 137 138 140 142 144 146 148 149 150 151 152 153 154 155 158 159 160 162 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 182 183 184 185 187 189 190 191 192 193]	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 27 28 29 30 32 33 34 35 36 37 38 39 40 41 42 43 46 47 48 49 50 51 52 53 56 58 59 60 61 62 63 64 65 66 67 68 69 70 71 72 73 74 75 76 77 78 79 81 82 84 85 86 87 88 89 90 91 92 93 94 95 96 97 98 99 100 101 103 104 105 106 107 108 110 111 112 113 115 117 119 120 121 122 123 124 125 126 127 128 129 130 131 132 133 135 137 138 139 140 142 144 146 148 149 150 151 152 153 154 155 158 159 160 162 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 182 183 184 185 187 189 190 191 192 193]	188364	550075	0.3	16
8	[22]	[16 17 18 20]	[14 15 16 18]	388	751	0.5	1
9	[22]	[52]	[46]	25	285	0.1	1
10	[23]	[188]	[188]	19	77	0.2	1

Table 5.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
11	[26 27 28 29 30 31 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47]	[1 2 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 19 20 29 30 31 32 34 36 37 38 39 40 41 43 44 45 46 47 48 49 50 51 52 53 54 55 56 57 58 59 62 63 64 66 67 68 69 70 71 72 73 74 75 76 77 79 80 81 82 83 84 85 86 87 88 89 90 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 114 115 116 117 118 119 120 124 126 127 128 129 130 131 132 133 134 135 136 137 138 140 141 142 143 144 146 147 148 151 152 153 154 155 158 160 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 182 183 184 185 187 188 190 192 193]	[1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16 17 18 27 28 29 30 32 33 34 35 36 37 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 56 58 59 60 62 63 64 65 66 67 68 69 70 71 72 73 75 76 77 78 79 80 81 82 83 84 85 86 89 90 91 92 93 94 95 96 97 98 99 100 101 102 103 104 105 106 107 108 109 110 111 112 113 117 119 120 121 122 123 124 125 126 127 128 129 130 131 132 134 135 137 138 139 140 141 142 143 144 146 147 148 151 152 153 154 155 158 160 162 163 164 165 166 167 168 169 170 171 172 173 174 175 176 177 178 179 180 182 183 184 185 187 188 190 192 193]	242016	735254	0.3	22
12	[27 28]	[41 43 46 47]	[37 39 42 43]	911	2892	0.3	2
13	[27]	[187]	[187]	48	99	0.5	1
14	[30 31 32 33 34 35 36 37 38 39 40]	[187 188 189 191 192 193]	[187 188 189 191 192 193]	5193	7481	0.7	11
15	[31]	[188]	[188]	111	114	1.0	1
16	[40]	[39 59]	[35 53]	105	643	0.2	1
17	[40 41]	[187 188]	[187 188]	150	468	0.3	2
18	[41 42]	[16 17]	[14 15]	138	562	0.2	2
19	[41]	[34]	[32]	18	362	0.1	1
20	[42]	[1 2 5 6]	[1 2 3 4]	391	1355	0.3	1
21	[42 43 44 45]	[192 193]	[192 193]	352	438	0.8	4
22	[43 44]	[52]	[46]	123	235	0.5	2
23	[43]	[63]	[59]	29	195	0.1	1
24	[44]	[5]	[3]	243	294	0.8	1
25	[44]	[14]	[12]	62	94	0.7	1
26	[44]	[17]	[15]	100	112	0.9	1
27	[44]	[99 100]	[92 93]	252	321	0.8	1
28	[44]	[166]	[166]	29	17	1.7	1
29	[44]	[186]	[186]	37	38	1.0	1

Table 5.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
30	[45]	[5]	[3]	33	238	0.1	1
31	[45]	[35]	[133]	27	301	0.1	1
32	[45 46]	[55 57 58]	[49 51 52]	264	396	0.7	2
33	[45]	[62]	[58]	30	199	0.2	1
34	[45]	[110]	[103]	18	45	0.4	1
35	[45]	[160]	[160]	15	46	0.3	1
36	[45]	[184]	[184]	48	28	1.7	1
37	[46]	[6]	[4]	53	101	0.5	1
38	[46]	[11]	[9]	27	88	0.3	1
39	[46]	[31]	[29]	37	68	0.5	1
40	[46]	[85]	[81]	64	55	1.2	1
41	[46]	[166]	[166]	28	30	0.9	1
42	[46]	[173]	[173]	95	110	0.9	1
43	[47]	[31]	[29]	35	47	0.7	1
44	[48]	[31]	[29]	29	65	0.4	1

Table 5.8: Final output showing disruptions of 4^{th} of September, 2017

The column named 'Disruption ID' numbers the spatially and temporally connected disruptions. The column 'Time' shows the time slice(s) where the disruptions occurred. The 'Node ID' column shows the node IDs affected by the disruptions, while the 'Link ID' column shows the corresponding link IDs. Note that for some disruptions that have a long duration and affect a large number of nodes and links, not all node IDs and link IDs can be seen in this table. The names of the stations represented by the node IDs can be found in Appendix D. The column 'Total passenger delay' shows the total passenger delay in minutes, calculated by summing the multiplication of the average passenger delay at that location with the passenger counts. The column 'counts' shows the total number of passengers that traversed the network elements during the time slice(s). Note that these are not individual passenger counts, a passenger that uses for example 5 network elements that are affected by the disruption is counted five times. The column 'Average delay' is the average passenger delay per affected network element during the time slice(s). This is calculated by dividing the total passenger delay by the passenger counts. The final column, 'Duration', shows the duration of the disruption in number of time slices. So, a duration of 1 means that the disruption lasted half an hour (or less).

5.4.9. Verification of detected disruptions within irregular week days

After the disruptions have been detected, the disruptions are verified. The only data available for verification is a disruption log file of WMATA. This disruption log file is not complete and is not completely trustworthy according to information from WMATA, but it will be interesting to see if the disruptions found are also found in the disruption log file and vice versa. It must be noted that the amount of disruptions found in this research is much larger than the number of disruptions in the log file. A summary of the findings and verification for all holidays and one track work day can be found in Table 5.9.

	04-09-17	**25-12-17**	**19-02-18**	**22-08-18**
Number of disruptions	44	139	34	29
Max total passenger delay	242,016	36,940	750,659	747,149
Avg total passenger delay	12,123	774	23,160	49,700
Max average passenger delay	1.72	8.28	5.38	4.92
Avg average passenger delay	0.60	0.87	0.98	1.08
Max duration (hours)	11	7	19	10.5
Avg duration (hours)	1.3	0.9	1.3	1.5
Max counts	735,254	165,310	1,811,513	2,234,329
Avg counts	33,288	2,846	54,844	159,419
Percentage from log file	42.9	34.3	35.7	39.1
Percentage in log file	4.2	3.4	5.5	10.7
Total passenger delay detected	533,406	107,592	787,434	1,441,294
Total passenger delay in log file	481,642	75,301	750,659	1,426,667
Percentage passenger delay in log file	90.3	70.0	95.3	99.0

Table 5.9: Summary detected disruptions in holidays and one track work day (22-08-18)

In Table 5.9, the number of disruptions, total passenger delay, average passenger delay, duration and passenger counts are calculated as in the previous section. Verification has been performed by checking for each (only spatially!) connected disruption, if one of the affected stations is found in the disruption log file for that time slice. This is done before temporally connecting the disruptions, to be able to verify per time slice. After temporally connecting the disruptions, information on which locations are affected at which specific time slice is lost. The row 'Percentage from log file' calculates which percentage of the disruptions in the log file is found in the output. The row 'Percentage in log file' calculates the percentage of disruptions found by the algorithm, that are found in the log file. It can be seen that this percentage is much lower, as the algorithm detects far more disruptions than are present in the log file. There are several explanations why this is the case. First, the log file is not completely reliable, and therefore does not capture all disruptions. Secondly, not all delays are caused by disruptions. Therefore, delays occur while no disruptions are noted in the log file. Lastly, the algorithm might overvalue the number of disruptions. Furthermore, the total passenger delay detected and the total passenger delay in log file are calculated. The total passenger delay detected represents the total passenger delay related to the disruptions detected by the algorithm. The total passenger delay in log file represents the total passenger delay detected by the algorithm, caused by disruptions that are verified in the log file. The percentage passenger delay in log file shows the percentage of the total passenger delay that is caused by disruptions verified by the log file. It can be seen that this percentage is quite large, above 90% for three of the four days. Only Christmas day (25-12-2017) has a lower percentage of 70%.

5.4.10. Disruption locations and causes analysis for irregular week days

After verifying the detected disruptions of the algorithm using the log file, it might also be interesting to look at the locations and causes of disruptions that are found by the algorithm, and also of disruptions that are not found. This is performed by checking for the four irregular days that are analysed in Section 5.4.9 the locations and causes of occurred disruptions according to the log file, and comparing these to the locations and causes of verified disruptions found by the algorithm. It was found that most of the disruptions that were found by the algorithm were caused by door malfunctions (5 times), personal relief (5 times), vandalism (4 times) and bodily fluids (4 times). The causes of disruptions that were most often not found by the algorithm are also door malfunctions (8 times), personal relief (8 times) and using a gap train (5 times). When looking at locations where disruptions were found, the most disruptions that were found by the algorithm took place at the Metro Center (4 times), Fort Totten (3 times) and Crystal City (3 times). Locations where disruptions

occurred that were most often not found by the algorithm are Stadium-Armory (5 times), Rosslyn (4 times) and West Falls Church (4 times).

5.5. Weekend days analysis

The same analysis has been done for weekend days, of which the results can be found in Appendix A. Interestingly, more irregular days were found in weekend days than in week days, while the number of weekend days was naturally smaller. Track works are usually planned during weekends, and events are more often organised during weekends. Therefore, weekends might follow a less regular pattern. Furthermore, for the weekend day clusters, no clear morning and afternoon peaks could be found, while these could be observed for most of the week day clusters. Another interesting finding is that, within the irregular week days, 3 of the 9 days were a holiday, while 5 other days were during the summer holiday and suffered from planned track work. For the irregular weekend days, less clear reasons could be found why these days were irregular. A summary of the findings and verification for a selection of irregular weekend days can be found in Table 5.10.

	07-10-17	**08-10-17**	**11-02-18**	**21-04-18**	**30-06-18**
Number of disruptions	118	178	61	72	42
Max total passenger delay	104,016	43,120	105,464	952,737	688,631
Avg total passenger delay	4,538	1,367	6,768	21,148	33,250
Max average passenger delay	3.44	5.35	3.59	5.74	3.55
Avg average passenger delay	0.46	0.79	0.93	1.11	1.04
Max duration (hours)	5	5	15	7.5	9.5
Avg duration (hours)	0.9	0.9	1.3	0.9	1.4
Max counts	511,900	257,593	241,148	1,053,219	1,408,879
Avg counts	18,902	6508	14,922	24,683	68,021
Percentage from log file	25.0	25.0	47.1	54.5	36.0
Percentage in log file	2.7	2.2	3.6	6.3	3.7
Total passenger delay detected	535,429	235,202	412,836	1,718,152	1,396,521
Total passenger delay in log file	215,657	88,583	151,318	1,174,209	1,282,893
Percentage passenger delay in log file	40.3	37.7	36.7	68.3	91.9

Table 5.10: Summary disruption detection in weekend days

Table 5.10 is calculated the same way as with the week days. It can be seen that overall, more disruptions are found in the irregular weekend days than in the irregular week days. This can be explained by the fact that weekend days have a less regular pattern than week days. Furthermore, the percentage of passenger delay that is verified in the log file is overall a bit lower than for the week days.

6

Conclusions and recommendations

In this chapter, the research is wrapped up by first reporting the key findings in section 6.1. After that, limitations are described in section 6.2 and possible applications are described in section 6.3. The chapter concludes with recommendations for future research in section 6.4.

6.1. Key findings

This study aimed to automatically detect disruptions offline, using smart card data. First, a literature research was performed. The studies that were most related to this one, used clustering to define average behaviour. This is necessary to define what disruptions are. To be able to use clustering to detect disruptions, training data and test data are required. Therefore, the data in this research was split up into two parts. The available data was of days from September 2017 until August 2018. Splitting up the data was done by taking days within the odd weeks and days within even weeks, to get a relatively even distribution of days. When for example the data would have been split up in the first half year of data and the second half year of data, the data set might be biased because of the relatively large number of public holidays in December and January (Christmas, New Years Eve) in the first half year of data and summer holidays (July, August) in the second half year. After splitting up the data into training data and test data, hierarchical clustering is applied to the training data. This results in clusters, which can be used to calculate the centroids of the clusters and to execute probabilistic classification. The probabilistic classification can be used to find the probability that a day from the test data is an irregular day. After that, the centroids of the clusters representing regular days are compared to the irregular days. Finally, the irregular delays within the irregular days are connected, both spatially and temporally to detect the disruptions. In this research, the main research question was formulated as follows:

How can disruptions and the related passenger delays in a metro network be detected automatically, using smart card data?

This section will first answer the corresponding sub questions, and conclude with an overarching answer on the main research question. The first sub question was as follows:

1. How can a distinction be made between a regular delay and a disruption?

To distinguish between a regular delay and a disruption, first it needs to be investigated which delays occur regularly in the network. This is done by clustering days that show a similar pattern. A study by Krishnakumari et al. (2019) provided input to this research. The average passenger delays were calculated for each track element (link, node or transfer station) for aggregated time slices (each half hour, so 48 slices per day). Taking direction into account, 193 links were defined for the 95 stations in the network. Therefore, also 193 nodes and 193 transfer stations were defined, giving the same stations different numbers depending on the connected link. The days were then represented as a vector with length (193*3*48), with the average passenger delay for each link, node and transfer station per time slice. Then, hierarchical clustering was applied on the vectors of the days within the training data set, using the Euclidean distance between days. The results showed a clear distinction between week days and weekend days. To avoid misclassification, the remainder of the analysis was done separately for both week days and weekend days. Hierarchical clustering using the

Euclidean distance was again applied to both the week days and weekend days, and at the same time the structural similarity between week and weekend days was investigated by using the SSIM index. The resulting clusters from both methods were finally combined. To get a comprehensible number of clusters, it was chosen for both week and weekend days to cut the dendrogram at five clusters. For the week days, one clear, large cluster was found, representing regular week days. This cluster also had the smallest delays. The other four clusters were significantly smaller, with much larger delays. For the clusters of the weekend days, it was less clear which clusters represent regular and irregular days. One cluster had relatively large delays, but was also the largest cluster. Therefore, it was assumed that this cluster represents regular weekend days. It might well be possible that large delays occur on a regular basis during the weekends, for example caused by track works that are most of the time planned during the weekends instead of during week days. Two other clusters were also assumed to represent regular days, because of their size and delay magnitude. After determining which clusters, created using the training data, represent regular and irregular days for both week days as well as weekend days, probabilistic classification is applied. The model is trained by the training data, using the cluster IDs and corresponding day vectors. After that, the probabilities to belong to each one of the clusters are calculated for the days within the test data. For the test data of the week days, it was found that 9 days had the highest probability of belonging to a cluster representing irregular days, or a low probability to belong to the cluster representing regular days. These 9 days were therefore considered to be irregular. For the test data of the weekend days, it was found that 16 days had the highest probability of belonging to a cluster representing irregular days, or a low probability of belonging to one of the clusters representing regular days. So 16 irregular weekend days were found in the test data.

2. How can disruptions be detected, using the distinction between a regular delay and a disruption?

After distinguishing between days that follow a regular delay pattern and days that do not, disruptions were detected within the days that do not follow a regular delay pattern. The first step was to compare the average passenger delay vector of the irregular day with centroid of the cluster it had the highest probability to belong to. After that, locations (link, nodes or transfer stations) that had a delay value above a certain threshold were found, to determine the locations affected by the disruptions. Thresholding is applied to reduce the sensitivity of the algorithm, as even regular days deviate from the centroid they belong to. After the locations were found, they were first connected in a spatial way, as delays within a certain time slice can propagate over the network. When two locations were connected, it was assumed that they are caused by the same disruption. After that, the detected disruptions are connected in a temporal way, as the same disruption can last longer than one time slice. After connecting disruptions in a temporal way, the duration of each disruption can be deduced. Furthermore, the locations that are affected by the disruptions are known.

3. How can detected disruptions be used to calculate the related passenger delays?

To calculate the passenger delays associated with the detected disruptions, the passenger counts and calculated average passenger delay are used. The total passenger delay is calculated by, for each disruption, summing the multiplication of the average passenger delay and passenger counts for each location. After that, the total passenger delay of one disruption is divided by the total passenger counts of this disruption to calculate the average passenger delay. This average passenger delay is the average passenger delay per network element, so not per passenger. The average passenger delay per passenger depends on the route a passenger takes, so this is the sum of the average passenger delay of each track element the passenger uses. Multiplying the average passenger delay per network element by the number of network elements affected by a disruption would not make sense, as, especially for large disruptions, it is unlikely that each passenger travels over all affected network elements.

How can disruptions and the related passenger delays in a metro network be detected automatically, using smart card data?

Disruptions in the WMATA network have been detected automatically offline, using smart card data as an input. The methods used are hierarchical clustering, machine learning using probabilistic classification, and algorithms to connect disruptions in both a spatial and temporal way. Interestingly, more irregular days were found in weekend days than in week days, while the number of weekend days was naturally smaller. This might be due to the fact that track works are usually planned during weekends, and events are more

often organised during weekends. Therefore, weekends might follow a less regular pattern. Furthermore, for the weekend day clusters, there were no clear morning and afternoon peaks, while these could be observed for most of the week day clusters. Another interesting fact is that, within the irregular week days, 3 of the 9 days were a holiday, while 5 other days were during the summer holiday and suffered from planned track work. For the irregular weekend days, less clear reasons could be found why these days were irregular. The most significant scientific contribution of this study, is the methodology that is used. Existing methods are combined to investigate the day-to-day regularity of delays. Hierarchical clustering is performed three times: using the Euclidean distance, the SSIM index and the former two combined. Especially using the SSIM index and treating the delay data per day as an image to cluster days is a new contribution.

6.2. Limitations

This study suffers from several limitations. First, the start location (station) of a disruption is not found. Finding this start location is very difficult, as delays in a public transport network propagate both upstream as well as downstream. Therefore, no start location has been pointed out. This would have been useful information for the network operator. Furthermore, the disruption log file that is used to verify disruptions is not complete and fully trustworthy. Even if the log file would have been complete and fully trustworthy, not all disruptions could have been verified, as not all disruptions lead to delays, and not all delays are caused by disruptions. Not all disruptions from the log file have been detected in this study, the percentage found was between 25.0% and 54.5%. On top of that, a lot more disruptions are detected than are present in the log file. Only between 2.2% and 10.7% of the detected disruptions can be found in the log file. This might be caused by the previously mentioned reason, namely that not all delays are caused by disruptions, but it might also be caused by a too sensitive algorithm. This could be reduced by raising the delay threshold, but in this study a higher threshold led to more scattered, disconnected disruptions. However, the percentage of total passenger delay of disruptions that are verified by the log file, is much higher. For week days, this percentage was higher than 90% for most of the days, and for weekend days between 36.7 and 91.9%. Another limiation of the study is that the calculated passenger counts per disruption are biased. The total number of passengers has been calculated per time slice, while the disruption does not necessarily have lasted the whole time slice. This means that the number of passengers affected by a disruption is probably overestimated. Hence for most of the disruptions, less passengers will have been affected than the output shows.

6.3. Applications

The outcomes of this study can be applied in multiple ways. WMATA can use the methods from this study to detect disruptions per day. At the end of the day, or in the morning, they can see if the day was regular or irregular. When the day was irregular, they can detect disruptions and find the locations that were affected. Not only WMATA could use these methods, any public transport operator that has access to tap-in and tap-out data and vehicle movements can calculate the average passenger delays and use these as an input to apply the methods to detect disruptions. For a bus network this might be more complex, as buses have in general a larger headway and lower passenger counts. On top of that, buses are more severely affected by external factors, such as traffic jams. Furthermore, this study can be used to find locations where disruptions occur most often, and total passenger delay associated with these disruptions. This can help public transport network operators to prioritise which locations to focus on to reduce passenger delays. The method in this study is not only interesting for public transport network operators, but could also be used for road networks. Speed data could be used to investigate the day-to-day regularity of road networks, to find disruptions that are caused by accidents instead of the regular traffic jams during peak periods.

6.4. Future research

In this section, some opportunities for future research will be proposed. First, it would be interesting to execute a study regarding the correlation between passenger delays and disruption types. Some disruption types might cause much larger passenger delays than others. Furthermore, instead of clustering per day, clustering per time slice could be applied. This way, the regularity of time periods per day can be investigated. The outcomes could be used to get much faster results: after the end of each time slice, the disruptions within that time slice can be detected. Lastly, an interesting study would be to predict delays, using tap-in and tap-out data of passengers. It could be investigated how delays propagate over time, based on previous delay. Furthermore, it would be interesting to see how the average passenger delay develops prior to disruptions

and to predict when a disruption is solved. On top of that, this study could be used as a step towards real-time disruption detection, for both public transport and road networks. When the day-to-day regularity (or the regularity of data from a smaller time period) has been investigated, real-time data that deviates from this regularity can indicate disruptions in the network. This real-time data could be smart card data from a public transport network or speed data from a road network.

Bibliography

Baswade, A. M. and Nalwade, P. S. (2013). Selection of initial centroids for k-means algorithm. *IJCSMC*, 2(7):161–164.

Briand, A., Toqué, F., Côme, E., and Oukhellou, L. (2018). Automatic detection of atypical events in a public transport network using smart card data. *Unpublished preprint*.

Camargos, R. C., Nietto, P. R., and do Carmo Nicoletti, M. (2016). Agglomerative and divisive approaches to unsupervised learning in gestalt clusters. In *International Conference on Intelligent Systems Design and Applications*, pages 35–44. Springer.

Djukic, T., Hoogendoorn, S., and Van Lint, H. (2013). Reliability assessment of dynamic od estimation methods based on structural similarity index. Technical report.

Fox, C. (2018). Use of gap trains on Line 1 has 'absolutely' improved service: TTC. https://www.cp24.com/news/use-of-gap-trains-on-line-1-has-absolutely-improved-service-ttc-1.4168967.

Handte, M., Iqbal, M. U., Wagner, S., Apolinarski, W., Marrón, P. J., Navarro, E. M. M., Martinez, S., Barthelemy, S. I., and Fernández, M. G. (2014). Crowd density estimation for public transport vehicles. In *EDBT/ICDT Workshops*, pages 315–322.

Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons, Inc.

Hartigan, J. A. and Wong, M. A. (1979). Algorithm as 136: A k-means clustering algorithm. *Journal of the Royal Statistical Society. Series C (Applied Statistics)*, 28(1):100–108.

Hemminki, S., Nurmi, P., and Tarkoma, S. (2013). Accelerometer-based transportation mode detection on smartphones. In *Proceedings of the 11th ACM conference on embedded networked sensor systems*, page 13. ACM.

Hladká, B. and Holub, M. (2018). Introduction to machine learning. *http://ufal.mff.cuni.cz/ hladka/2018/docs/lec.svm-roc.2018-11-21.pdf*.

Jespersen-Groth, J., Potthoff, D., Clausen, J., Huisman, D., Kroon, L., Maróti, G., and Nielsen, M. N. (2009). Disruption management in passenger railway transportation. In *Robust and online large-scale optimization*, pages 399–421. Springer.

Ji, T., Fu, K., Self, N., Lu, C.-T., and Ramakrishnan, N. (2018). Multi-task learning for transit service disruption detection. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 634–641. IEEE.

Jiang, F., Yuan, J., Tsaftaris, S. A., and Katsaggelos, A. K. (2011). Anomalous video event detection using spatiotemporal context. *Computer Vision and Image Understanding*, 115(3):323–333.

Jobson, J. D. (1992). *Applied multivariate data analysis: volume II: Categorical and Multivariate Methods*. Springer Science & Business Media.

Kepaptsoglou, K. and Karlaftis, M. G. (2010). A model for analyzing metro station platform conditions following a service disruption. In *13th International IEEE Conference on Intelligent Transportation Systems*, pages 1789–1794. IEEE.

Kittelson & Associates, et al. (2003). *Transit capacity and quality of service manual*. Number 100. Transportation Research Board.

Krishnakumari, P., Cats, O., and Van Lint, H. (2019). Day-to-day and seasonal regularity of network passenger delay for metro networks. *Unpublished manuscript*.

Lopez, C., Leclercq, L., Krishnakumari, P., Chiabaut, N., and Van Lint, H. (2017). Revealing the day-to-day regularity of urban congestion patterns with 3d speed maps. *Scientific Reports*, 7(1):14029.

Mathworks (n.d.). Dendrogram. *https://nl.mathworks.com/help/stats/dendrogram.html*.

Musa, A. and Eriksson, J. (2012). Tracking unmodified smartphones using wi-fi monitors. In *Proceedings of the 10th ACM conference on embedded network sensor systems*, pages 281–294. ACM.

Noursalehi, P. and Koutsopoulos, H. N. (2016). Real-time predictive analytics for improving public transportation systems' resilience. *ArXiv*, abs/1609.09785.

Parsa, A. B., Taghipour, H., Derrible, S., and Mohammadian, A. K. (2019). Real-time accident detection: coping with imbalanced data. *Accident Analysis & Prevention*, 129:202–210.

Pender, B., Currie, G., Delbosc, A., and Shiwakoti, N. (2012). Planning for the unplanned: An international review of current approaches to service disruption management of railways. In *35th Australian Transport Research Forum, Perth, Australia*.

Rasmussen, C. and Williams, C. (2006). *Gaussian Processes for Machine Learning*. the MIT Press.

SciPy (2019). Scipy cluster hierarchy linkage. https://docs.scipy.org/doc/scipy/reference/generated/scipy.cluster.hierarchy.linkage.html.

Shalizi, C. (2009). Distances between clustering, hierarchical clustering. *Lecture notes*.

Sharma, A. (2018). Most popular clustering algorithms used in machine learning. https://www.analyticsindiamag.com/most-popular-clustering-algorithms-used-in-machine-learning/.

Song, B. and Wynter, L. (2017). Real-time public transport service-level monitoring using passive wifi: a spectral clustering approach for train timetable estimation. *arXiv preprint arXiv:1703.00759*.

Starkweather, J. and Moske, A. K. (2011). Multinomial logistic regression. *Consulted page at September 10th: http://www. unt. edu/rss/class/Jon/Benchmarks/MLR_JDS_Aug2011. pdf*, 29:2825–2830.

Tonnelier, E., Baskiotis, N., Guigue, V., and Gallinari, P. (2018). Anomaly detection in smart card logs and distant evaluation with twitter: a robust framework. *Neurocomputing*, 298:109–121.

Uniman, D. L. (2009). *Service reliability measurement framework using smart card data: Application to the london underground*. PhD thesis, Massachusetts Institute of Technology.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

WMATA (2019a). History. https://www.wmata.com/about/history.cfm.

WMATA (2019b). Metro's improved 'rush hour promise' begins tomorrow. https://www.wmata.com/about/news/2019-Rush-Hour-Promise.cfm.

WMATA (2019c). Wmata metro snapshot. https://www.wmata.com/about/upload/2019-Metro-Snapshot-Fact-Sheet.pdf.

WMATA (2019d). Wmata metrorail pocket guide. https://www.wmata.com/rider-guide/new-riders/upload/pocket-guide-English.pdf.

Yim, O. and Ramdeen, K. T. (2015). Hierarchical cluster analysis: comparison of three linkage measures and application to psychological data. *The quantitative methods for psychology*, 11(1):8–21.

A

Weekend days analysis

A.1. Hierarchical clustering of weekend days based on Euclidean distance

In Figure A.1, the results of the hierarchical clustering, using the Euclidean distance between days, for the weekend days of the odd weeks are shown. It is chosen to cut the tree at four clusters, as adding another cluster would split up the already small cluster 2 into two clusters, one containing 4 days and one containing one day.

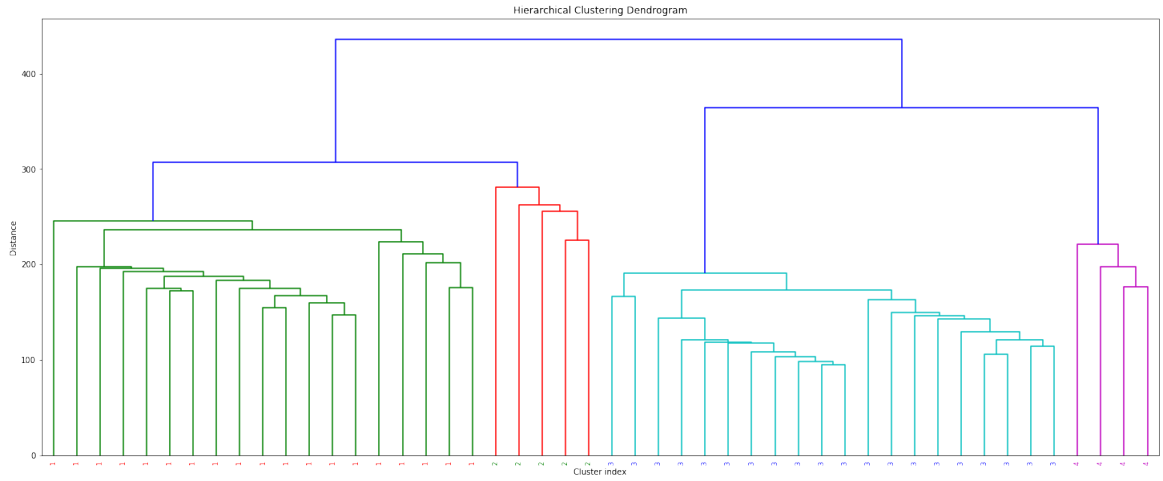


Figure A.1: Dendrogram weekend days solely based on delay values

An overview of the cluster sizes can be found in the truncated dendrogram in Figure A.2.

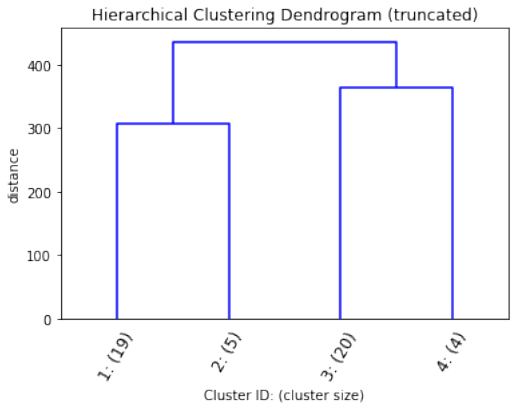
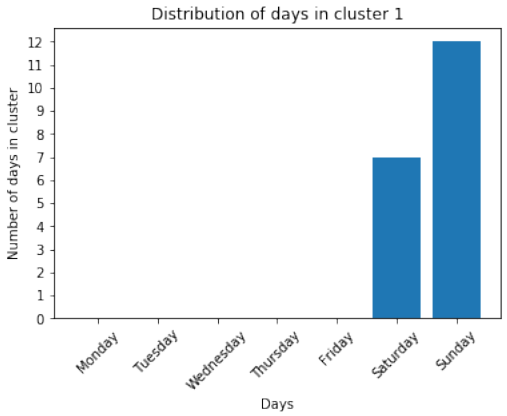


Figure A.2: Truncated dendrogram weekend days solely based on delay values

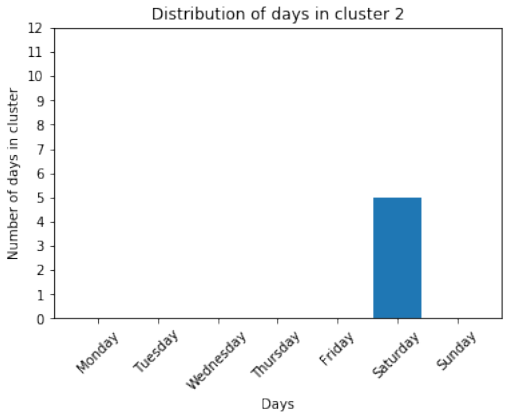
Figure A.2 shows that there are two larger clusters, cluster 1 containing 19 days and cluster 3 containing 20 days. The two smaller clusters are cluster 2 and 4 that contain 5 and 4 days respectively.

A.1.1. Distribution of days for clusters of weekend days based on Euclidean distance

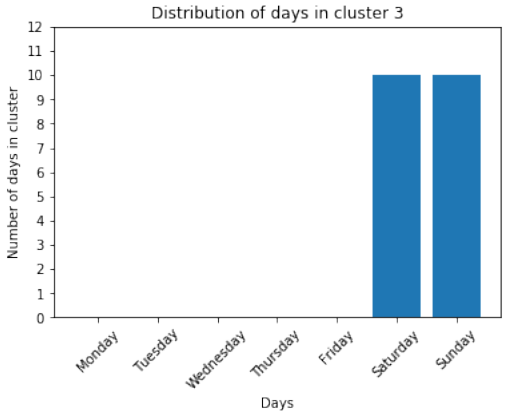
In this section, the distribution of days for each cluster, solely based on the Euclidean distance between days, will be shown.



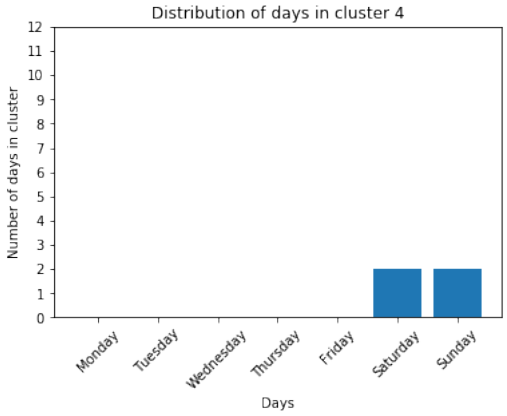
(a) Distribution of weekend days in cluster 1 solely based on delay values



(b) Distribution of weekend days in cluster 2 solely based on delay values



(c) Distribution of weekend days in cluster 3 solely based on delay values



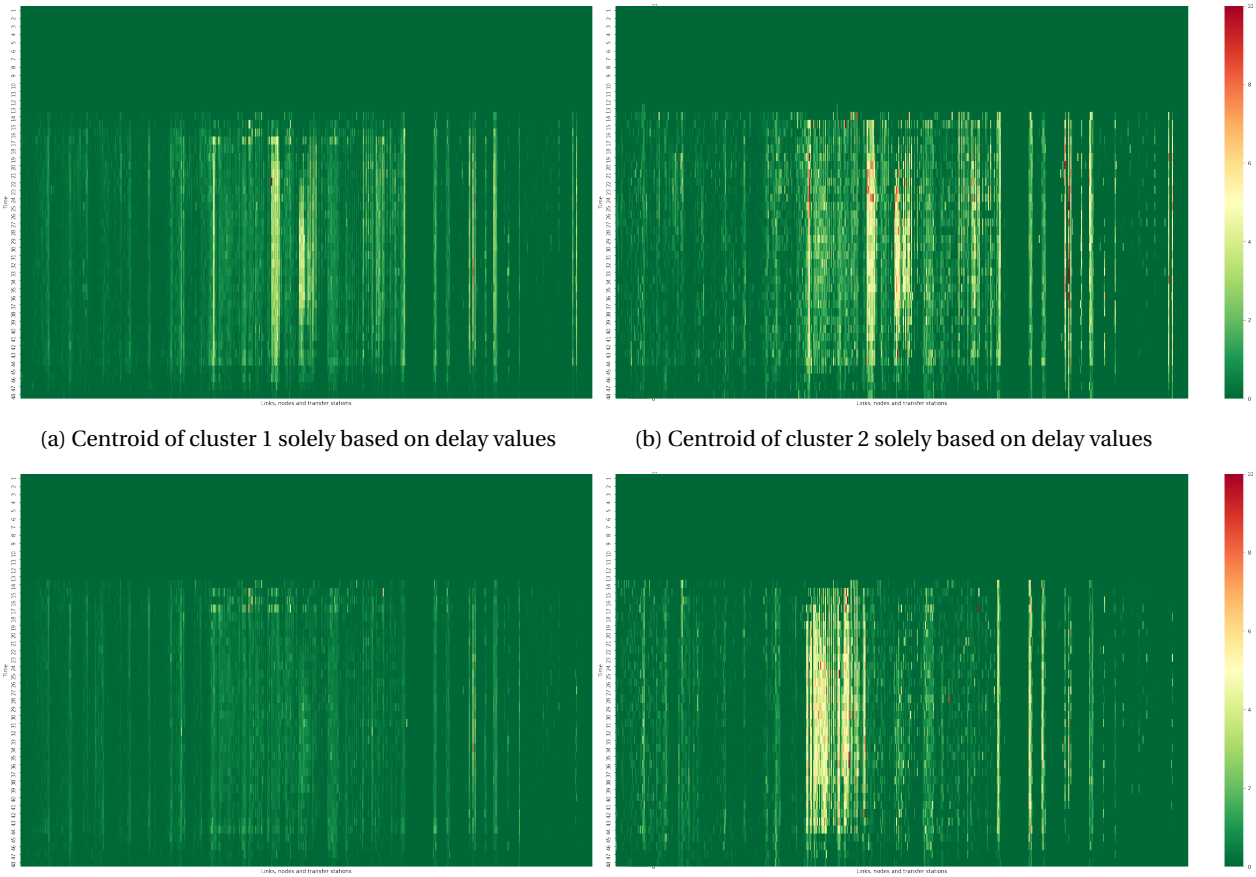
(d) Distribution of weekend days in cluster 4 solely based on delay values

Figure A.3: Distribution of weekend days in clusters solely based on delay values

Figure A.3 shows that cluster 1 contains both Saturdays and Sundays, but the number of Sundays is larger. Cluster 2 contains only Saturdays. Cluster 3 and 4 both have an equal number of Saturdays and Sundays, but cluster 3 is much larger.

A.1.2. Centroids of clusters based on Euclidean distance for weekend days

In this section, the centroids of weekend day clusters solely based on Euclidean distance will be shown. Again, the centroids show the time slices on the y-axis and the links, nodes and transfer stations respectively on the x-axis.



(a) Centroid of cluster 1 solely based on delay values

(b) Centroid of cluster 2 solely based on delay values

(c) Centroid of cluster 3 solely based on delay values

(d) Centroid of cluster 4 solely based on delay values

Figure A.4: Centroids of clusters of weekend days solely based on delay values

Figure A.4 shows that cluster 2 and 4 both contain quite large delays. Cluster 1 has slightly smaller delays, while cluster 3 has the smallest delays. No real morning or evening peak can be derived from these figures.

A.2. Hierarchical clustering of weekend days based on SSIM index

In Figure A.5, the results of the hierarchical clustering for the weekend days of the odd weeks are shown, based on the SSIM index. It is again chosen to cut the tree at five clusters, as this is an understandable number of clusters.

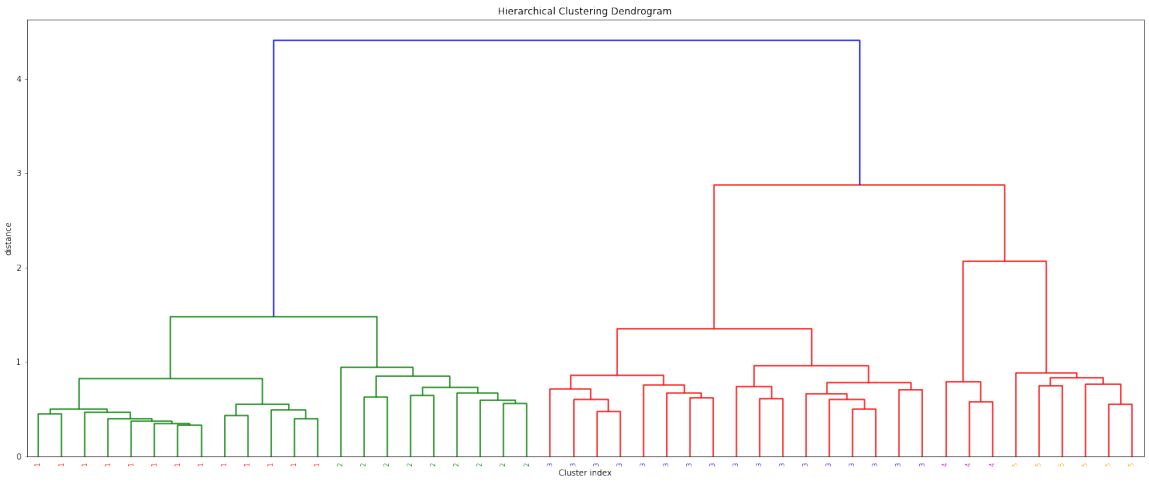


Figure A.5: Dendrogram weekend days based on SSIM index

An overview of the cluster sizes can be found in the truncated dendrogram in Figure A.6.

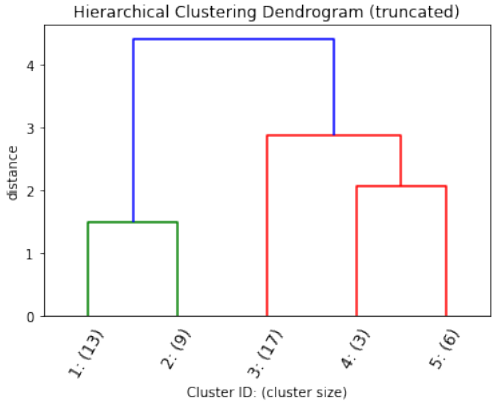
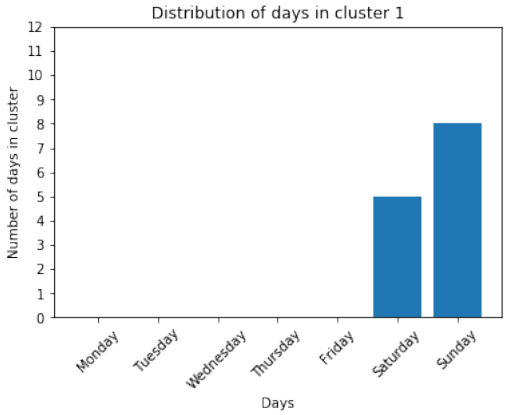


Figure A.6: Truncated dendrogram weekend days based on SSIM index

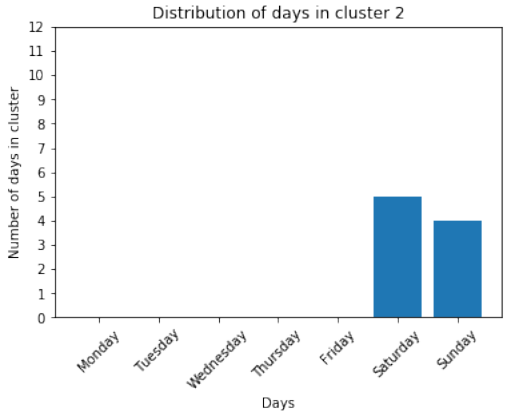
Figure A.6 shows that there are two larger clusters, cluster 1 containing 13 days and cluster 3 containing 17 days. Cluster 2 consists of 9 days, while cluster 4 consists of 3 days and cluster 5 contains 6 days.

A.2.1. Distribution of days for clusters of weekend days based on SSIM index

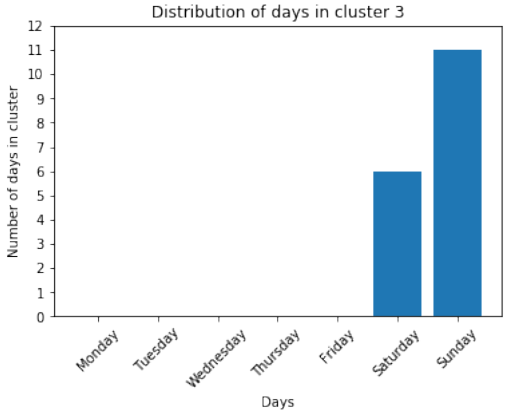
In this section, the distribution of days for each cluster, based on the SSIM index, will be shown.



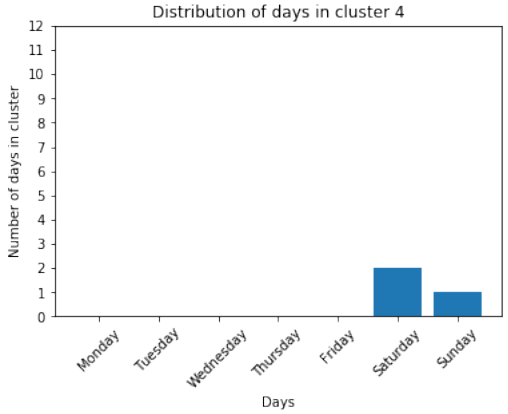
(a) Distribution of weekend days in cluster 1 based on SSIM index



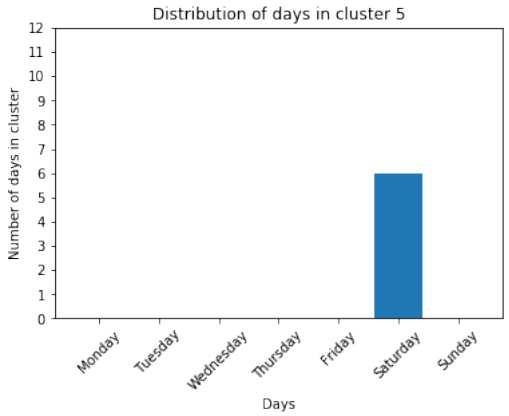
(b) Distribution of weekend days in cluster 2 based on SSIM index



(c) Distribution of weekend days in cluster 3 based on SSIM index



(d) Distribution of weekend days in cluster 4 based on SSIM index



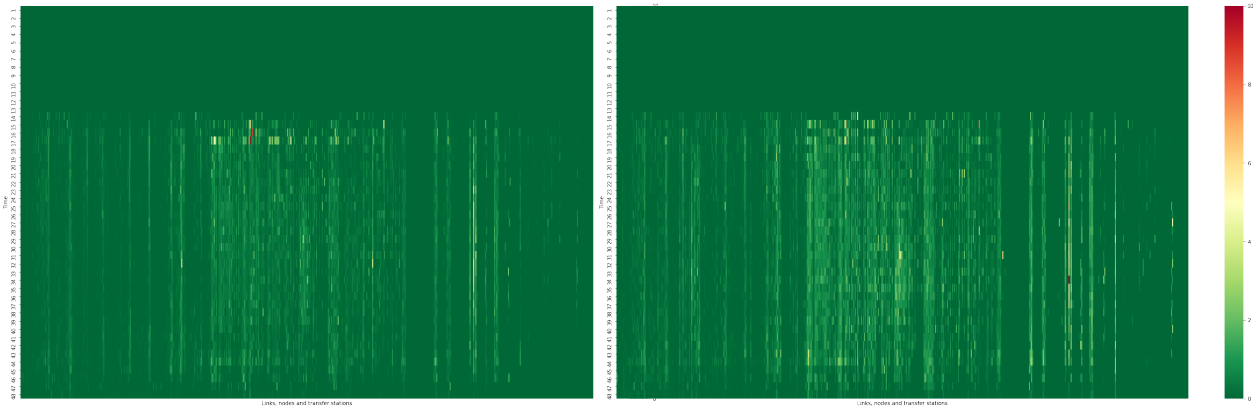
(e) Distribution of weekend days in cluster 5 based on SSIM index

Figure A.7: Distribution of weekend days in clusters based on SSIM index

Figure A.7 shows that the first four clusters all contain both days. Cluster 1 and 4 contain more Sundays than Saturdays, and cluster 2 and 4 vice versa. Cluster 5 only consists of Saturdays.

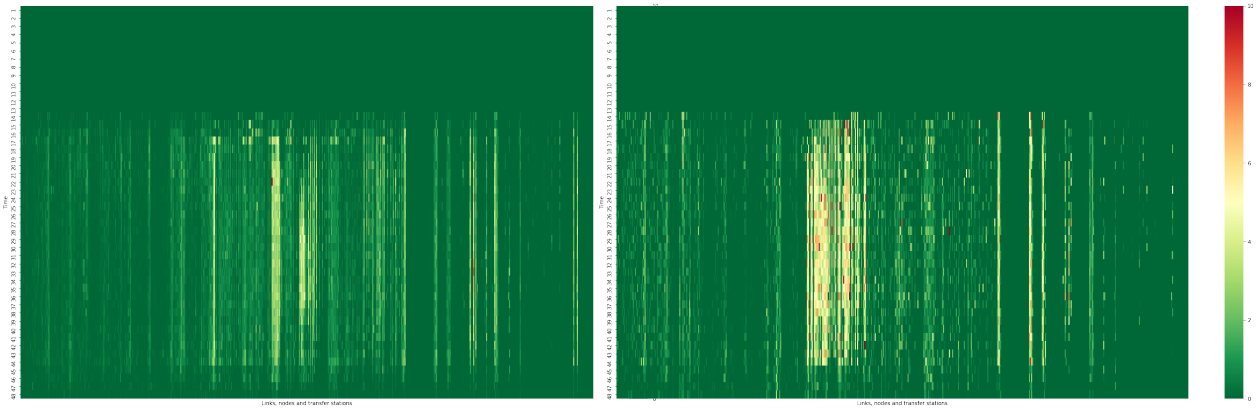
A.2.2. Centroids of clusters based on SSIM index for weekend days

In this section, the centroids of clusters based on the SSIM index will be shown.



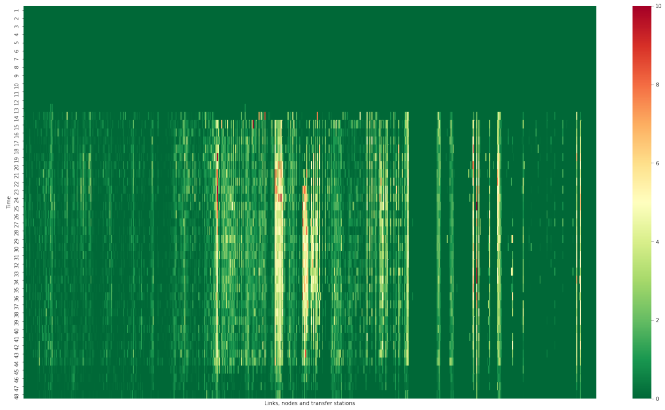
(a) Centroid of cluster 1 based on SSIM index

(b) Centroid of cluster 2 based on SSIM index



(c) Centroid of cluster 3 based on SSIM index

(d) Centroid of cluster 4 based on SSIM index



(e) Centroid of cluster 5 based on SSIM index

Figure A.8: Centroids of clusters of weekend days based on SSIM index

Figure A.8 shows that cluster 4 contains the largest delays. This is also the smallest cluster, consisting of only three days. Furthermore, cluster 5 and 3 also show quite large delays. Cluster 1 and 2 show the smallest delays.

A.3. Hierarchical clustering of weekend days based on both delay values and SSIM index

In Figure A.9, the results of the hierarchical clustering for the weekend days of the odd weeks are shown, based on both the delay values and the SSIM index. It is again chosen to cut the tree at five clusters.

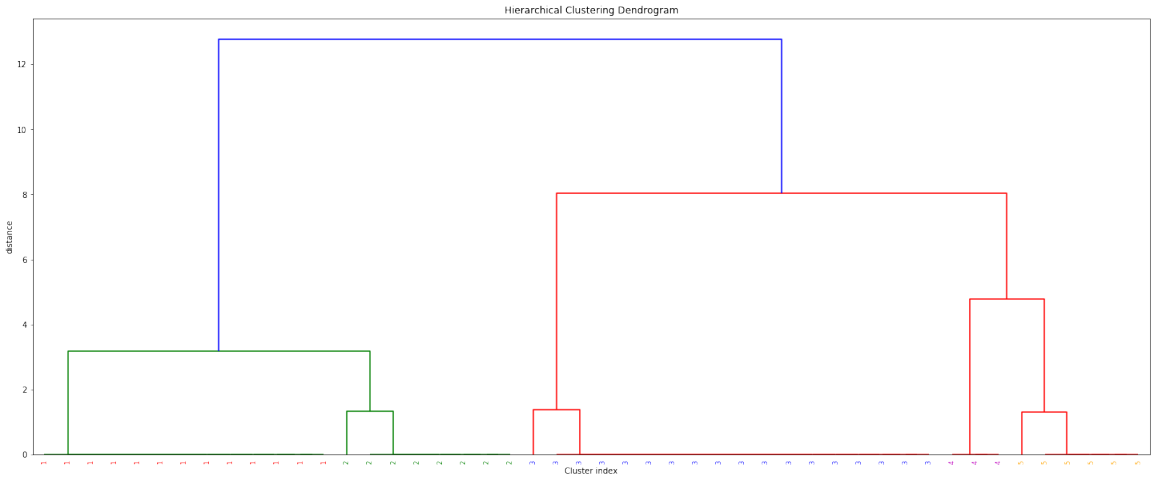


Figure A.9: Dendrogram weekend days based on both delay values and SSIM index

An overview of the cluster sizes can be found in the truncated dendrogram in Figure 5.17.

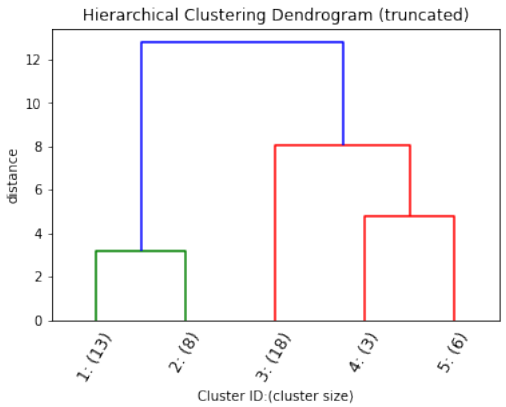
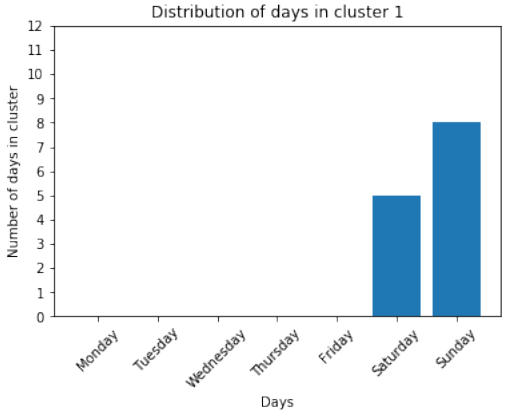


Figure A.10: Truncated dendrogram weekend days based on both delay values and SSIM index

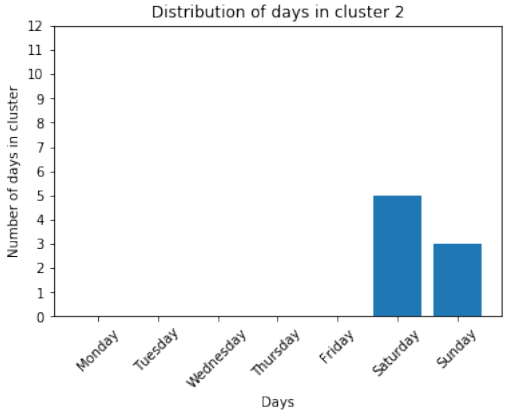
Figure A.10 shows that there are two larger clusters, cluster 1 containing 13 days and cluster 3 containing 18 days. Cluster 2 consists of 8 days, cluster 5 of 6 days and cluster 4 of 3 days.

A.3.1. Distribution of days for clusters of weekend days based on delay values and SSIM index

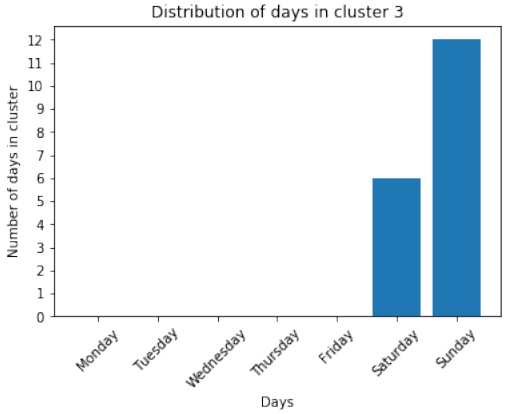
In this section, the distribution of days for each cluster, based on both the delay values and the SSIM index, will be shown.



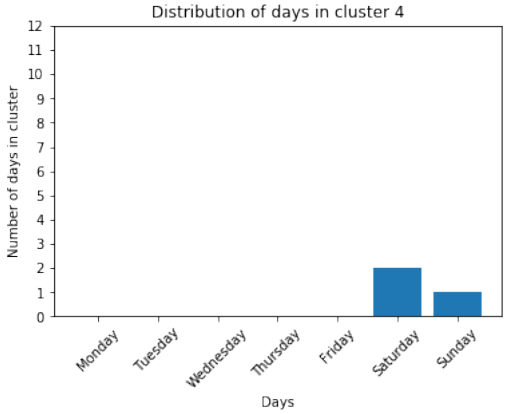
(a) Distribution of weekend days in cluster 1 based on delay values and SSIM index



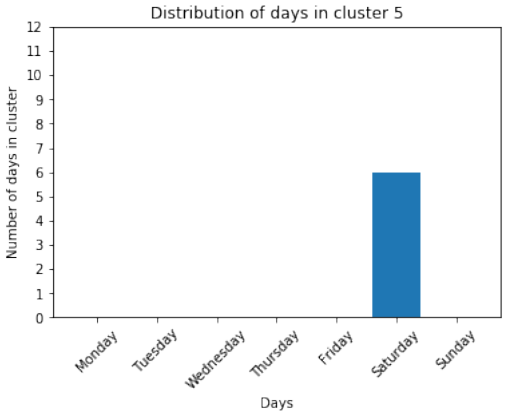
(b) Distribution of weekend days in cluster 2 based on delay values and SSIM index



(c) Distribution of weekend days in cluster 3 based on delay values and SSIM index



(d) Distribution of weekend days in cluster 4 based on delay values and SSIM index



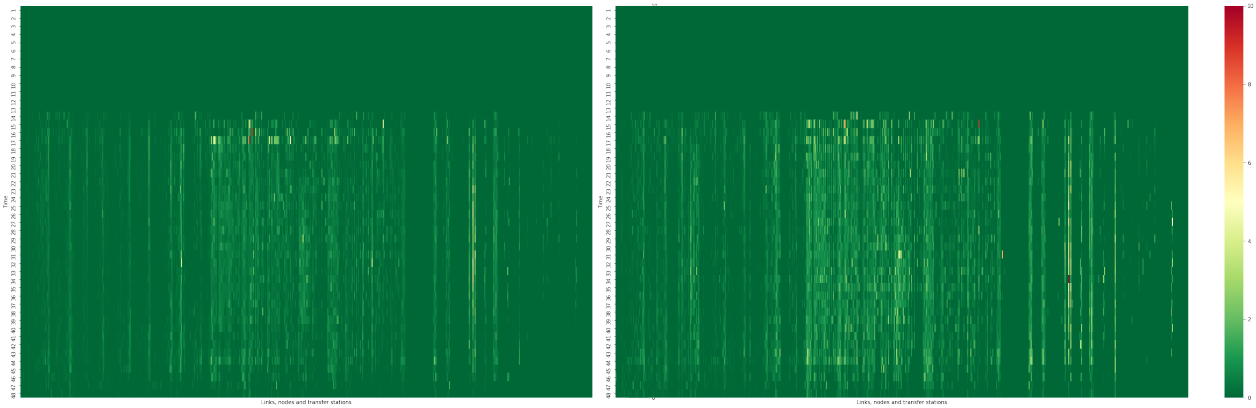
(e) Distribution of weekend days in cluster 5 based on delay values and SSIM index

Figure A.11: Distribution of weekend days in clusters based on delay values and SSIM index

Figure A.11 shows that the first four clusters all contain both days. Cluster 1 and 3 contain more Sundays than Saturdays and cluster 2 and 4 vice versa. Cluster 5 contains only Saturdays.

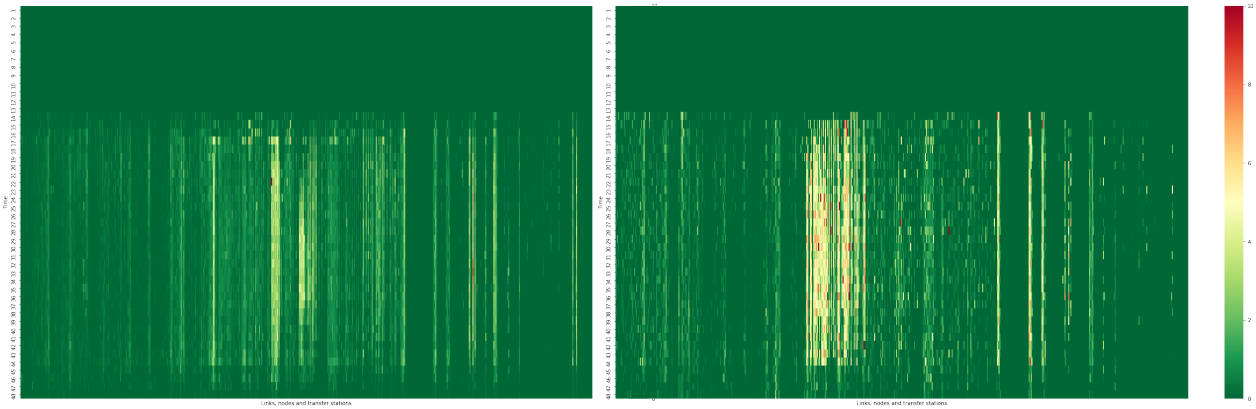
A.3.2. Centroids of clusters based on Euclidean distance and SSIM index for weekend days

In this section, the weekend days centroids of clusters based on the Euclidean distance and SSIM index will be shown.



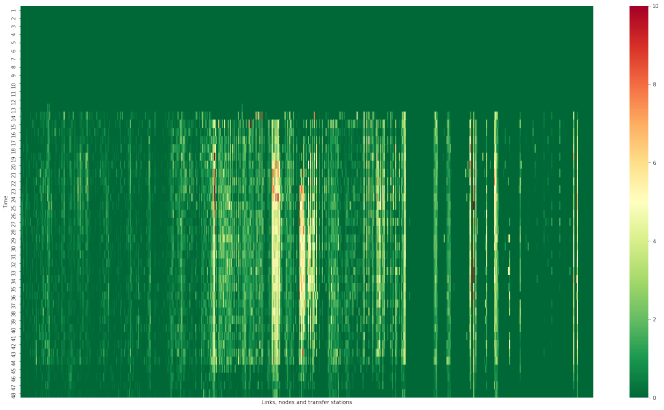
(a) Centroid of cluster 1 based on Euclidean distance and SSIM index

(b) Centroid of cluster 2 based on Euclidean distance and SSIM index



(c) Centroid of cluster 3 based on Euclidean distance and SSIM index

(d) Centroid of cluster 4 based on Euclidean distance and SSIM index



(e) Centroid of cluster 5 based on Euclidean distance and SSIM index

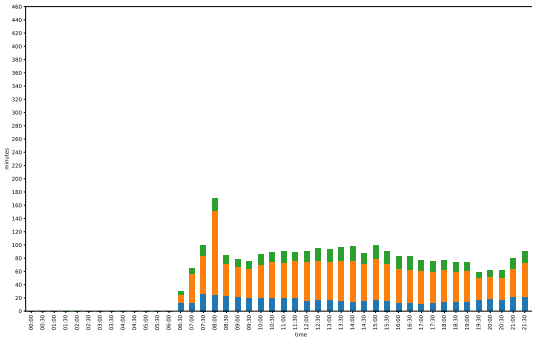
Figure A.12: Centroids of clusters of weekend days based on Euclidean distance and SSIM index

Figure A.12 shows that the centroid of cluster 4 contains the largest delays, followed by cluster 5. Cluster 3 and 4 contain also quite large delays. Cluster 1 and 2 show the smallest delays. The same way as with the week days, a histogram of the stacked link, node and transfer delays for each time slice is created for each

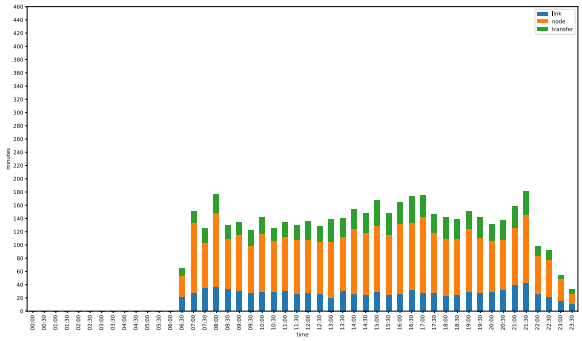
cluster, to get a better insight in which kind of delays (node, link or transfer delays) occur at what time slices.

A.3.3. Histograms of the stacked link, node and transfer delays for each cluster

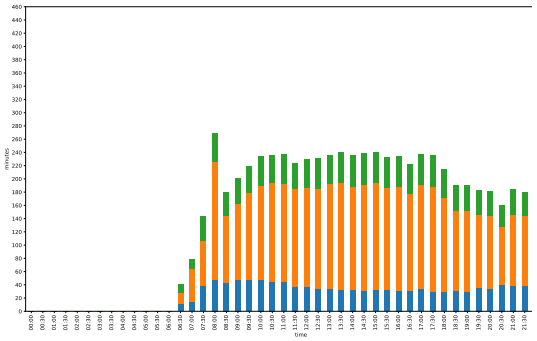
In this section, the histograms of the stacked link, node and transfer delays are shown for each cluster.



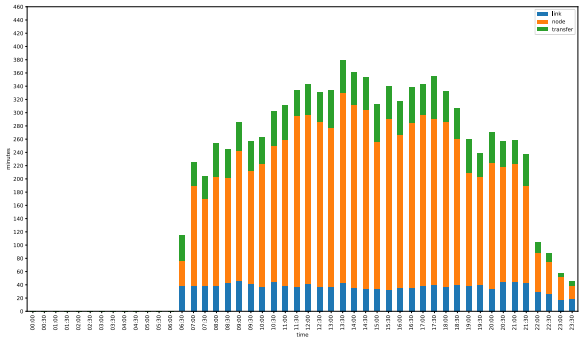
(a) Stacked histogram for link, node and transfer delays per time slice of cluster 1



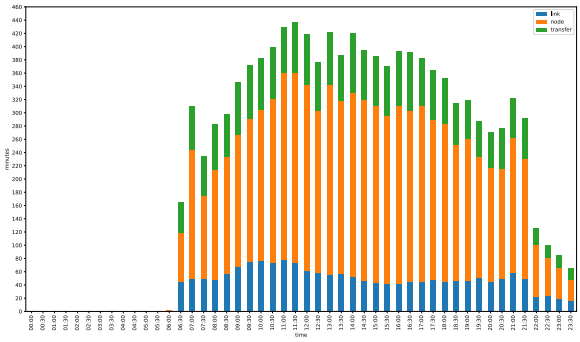
(b) Stacked histogram for link, node and transfer delays per time slice of cluster 2



(c) Stacked histogram for link, node and transfer delays per time slice of cluster 3



(d) Stacked histogram for link, node and transfer delays per time slice of cluster 4



(e) Stacked histogram for link, node and transfer delays per time slice of cluster 5

Figure A.13: Centroids of clusters of weekend days based on delay values and SSIM index

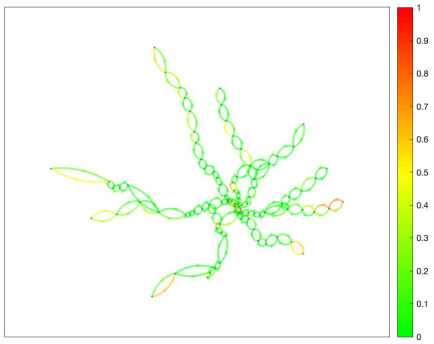
Figure A.13 shows that overall, the node delays are the largest of all three delays. Cluster 1 shows the largest delays between 8 am and 8:30 am (time slice 17). Cluster 2 has no real distinct peak, but the largest delays can be seen between 9:30 and 10 pm (slice 44). The peak at cluster 3 takes place between 8 am and

8:30 am (slice 17). Cluster 4 shows the largest delay between 1:30 and 2 pm (slice 28), while the largest delay at cluster 5 occurs between 11:30 am and noon (slice 24). After determining the time slices that contain the largest delays, the locations where these delays occur will again be investigated. In the next section, graphs will be shown of certain time slices to indicate these locations.

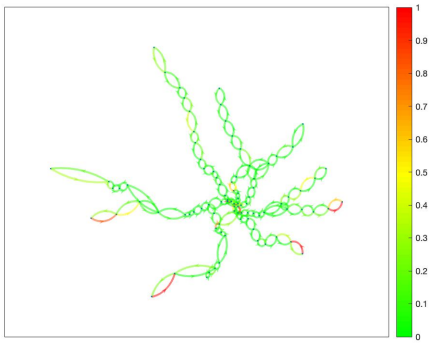
A.3.4. Graphs of the network indicating locations where delays occur at certain time slices

In this section, graphs will be shown that indicate the locations where certain types of delay occur for each cluster.

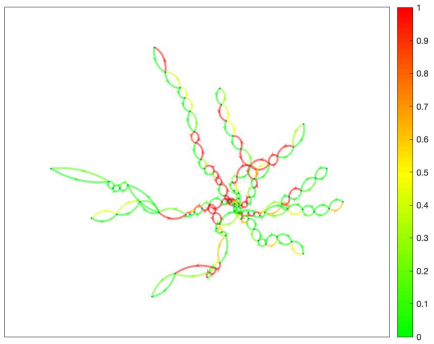
In cluster 1, the largest delays occur between 8 am and 8:30 am. Therefore, this time slice and the time slice after that will be investigated. Separate graphs are made for each type of delay, the node, link and transfer delays.



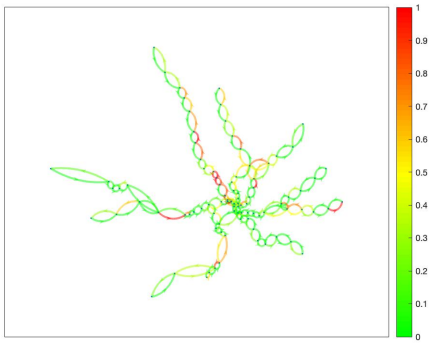
(a) Network graph representing link delays between 8 am and 8:30 am for cluster 1



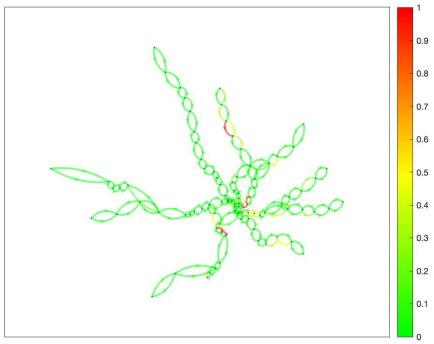
(b) Network graph representing link delays between 8:30 am and 9 am for cluster 1



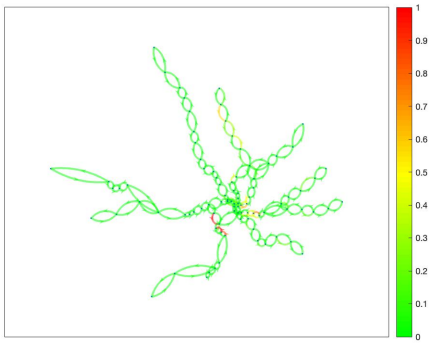
(c) Network graph representing node delays between 8 am and 8:30 am for cluster 1



(d) Network graph representing node delays between 8:30 am and 9 am for cluster 1



(e) Network graph representing transfer delays between 8 am and 8:30 am for cluster 1



(f) Network graph representing transfer delays between 8:30 am and 9 am for cluster 1

Figure A.14: Network graphs representing link, node and transfer delays between 8 am and 9 am for cluster 1

Figure A.14 shows that link delays between 8 and 8:30 am mostly occur on the outskirts of the network. It can be seen that in the next time slice, the link delays in the West, South and East get more severe. The node delays between 8 am and 8:30 are widely spread over the network. In the next time slice, it can be seen that most of the node delays are reduced. Furthermore, it can be seen that the largest transfer delays between 8 and 8:30 am can be found in the North and South region. In the adjacent time slice, the transfer delays in the North have been largely reduced, however there are still transfer delays in the South. These graphs have been made for all clusters and all time slices with peak delays. These can be found in Appendix C. Summaries of the findings of these graphs for each cluster can be found in Tables A.1, A.2, A.3, A.4 and A.5.

Delay peak time	Locations link delay	Locations node delay	Locations transfer delay
8 - 9 am	North, West, South, East	North, South, Centre	North, Centre

Table A.1: Delay peak time and locations cluster 1 for weekend days

Delay peak time	Locations link delay	Locations node delay	Locations transfer delay
9:30 - 10:30 pm	North, West, South, East	North, West, South	North, Centre

Table A.2: Delay peak time and locations cluster 2 for weekend days

Delay peak time	Locations link delay	Locations node delay	Locations transfer delay
8 - 9 am	West, East	Whole network	North, West, Centre

Table A.3: Delay peak time and locations cluster 3 for weekend days

Delay peak time	Locations link delay	Locations node delay	Locations transfer delay
1:30 - 2:30 pm	North, South, East, Centre	North, Centre	North, West, Centre

Table A.4: Delay peak time and locations cluster 4 for weekend days

Delay peak time	Locations link delay	Locations node delay	Locations transfer delay
11:30 am - 12:30 pm	North, West, South, East, Centre	Whole network	North, West, Centre

Table A.5: Delay peak time and locations cluster 5 for weekend days

Table A.6 shows a summary for each cluster, including the number of days within each cluster, the total link, node and transfer delay, the total of these three delays and the time period where the largest delay occurs.

	Cluster 1	Cluster 2	Cluster 3	Cluster 4	Cluster 5
Number of days	13	8	18	3	6
Total link delay (min)	585	977	1152	1298	1762
Total node delay (min)	1659	2848	4141	6587	7155
Total transfer delay (min)	488	891	1275	1474	2253
Total delay (min)	2731	4716	6568	9359	11170
Period where largest delay occurs	Morning	Evening	Morning	Afternoon	Afternoon

Table A.6: Summary of each cluster for the weekend days

A.4. Probabilistic classification for weekend days

The centroids of each cluster based on both delay values and the SSIM index are used to detect disruptions. As with the weekdays, to determine whether a day that has a probability to belong to a cluster is irregular or not, it must be determined whether the cluster represents irregular days or regular days. This is less clear with the weekend days clusters than with the week day clusters. Cluster 3 has relatively large delays, but is also the largest cluster. Therefore, this cluster is considered to represent regular days. Cluster 1 is a bit smaller, but has much smaller delays and is therefore considered to be representing regular days. Cluster 2 is a small cluster, but has less delays than cluster 3. Cluster 4 and 5 are the smallest clusters with the largest delays, so these clusters are considered to represent irregular days. As with the week days, with probabilistic classification, the probabilities of belonging to each of the clusters are calculated for each day. While with the week days, a probability of below 30% and below 70% both led to the same number of irregular days, this is not the case for the weekend days. When a probability of 30% is used, 3 irregular days are found, while when a probability of 70% is used, 28 irregular days are found. It is chosen to use the average of these probability, 50%, of belonging to cluster 1, 2 or 3, which leads to 16 irregular days. This is a lot more than the number of irregular weekdays that were found, while the number of weekend days that are analysed is much smaller. This might be caused by weekends following a less regular pattern, because more events and track work take place during weekends. The dates of the irregular days including a possible explanation why this day is irregular can be found in Table 5.7. Note that when a box is filled with '-', this does not necessarily mean that there was no event or track work, but that this is not known.

Date	**Event**	**Track work**
07-10-17	Columbus day weekend	-
08-10-17	Columbus day weekend	-
16-12-17	Washington Nationals Winterfest	-
14-01-18	-	-
11-02-18	-	-
24-03-18	Protest march against gun violence; "March of our lives" - 200,000-800,000 attendants	-
21-04-18	-	-
05-05-18	-	-
06-05-18	-	-
02-06-18	-	-
30-06-18	Protest march against immigration policies; "Families belong together" - more than 30,000 attendants	-
14-07-18	-	-
11-08-18	-	Yes
12-08-18	White supremacist rally; "Unite the right 2" - thousands attendants	Yes
25-08-18	-	Yes
26-08-18	-	Yes

Table A.7: Irregular weekend dates and possible explanation why these dates are irregular

The white supremacist rally on 12-08-18 is especially interesting, as WMATA closed the Vienna station and only allowed white supremacist supporters, police and press in. They travelled from Vienna to Foggy Bottom. There were only 20-30 supporters, however thousands of counter-demonstrators appeared. However, this has not increased the number of disruptions, as it turns out that this day is irregular because it has less disruptions than its cluster it has the highest probability to belong to. Now that the irregular days have been determined, the disruptions need to be found within these days. Therefore, the centroid from the cluster with the highest probability is subtracted from the delay matrix of the irregular day. This results in a matrix that consists of disruptions when compared to an average day of that class.

A.5. Disruption detection in weekend days

To show the link, node and transfer delay of the resulting matrix consisting of disruptions when compared to the average of the cluster it belongs to, a stacked histogram is made. This histogram from the first irregular weekend day, 7^{th} of October, 2017 is shown in Figure A.15. This day has the highest probability of belonging to cluster 1, therefore this day is compared to the centroid of cluster 1.

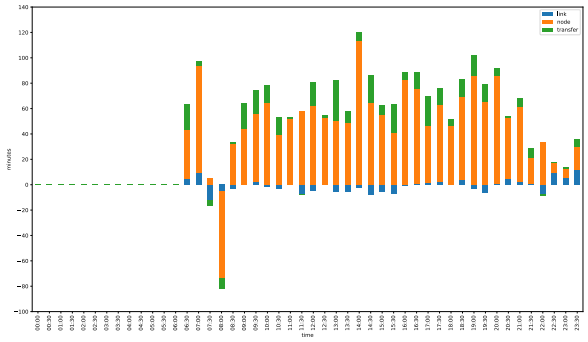
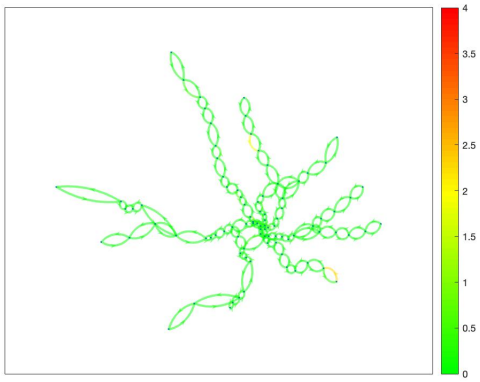
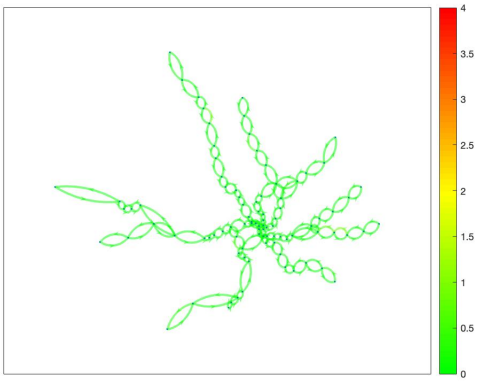


Figure A.15: Stacked histogram of link, node and transfer delays of 07-10-17

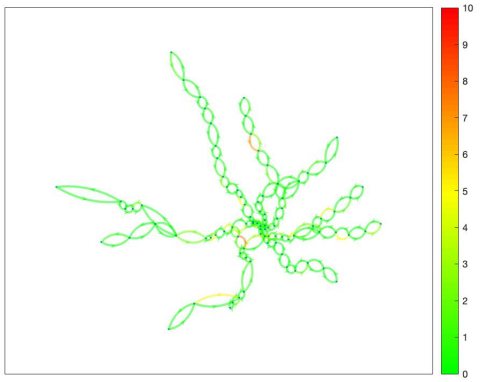
It can be seen that between 7:30 am and 8:30 am (time slice 16 and 17) the delays are smaller than in centroid 1. However, the delays for the other time slices are higher. On top of that, most of the link delays are smaller than in centroid 1. The largest delay occurs between 2 pm and 2:30 pm (time slice 29), therefore network graphs are made of this time slice and the successive time slice.



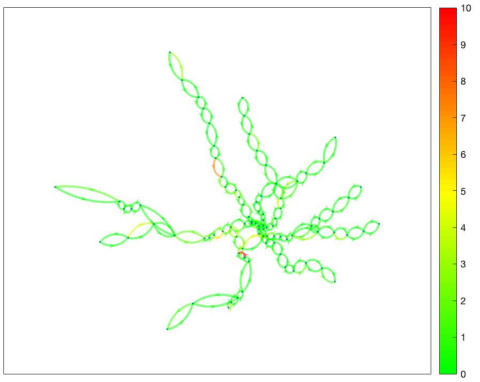
(a) Network graph representing link delays between 2 pm and 2:30 pm for 07-10-17



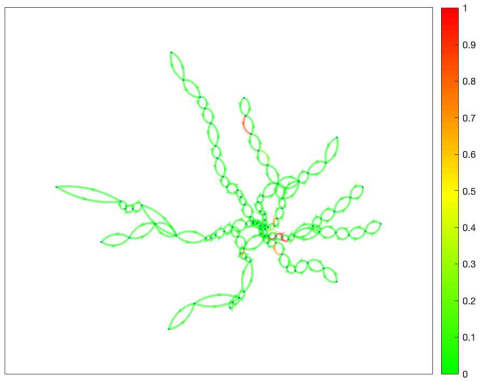
(b) Network graph representing link delays between 2:30 pm and 3 pm for 07-10-17



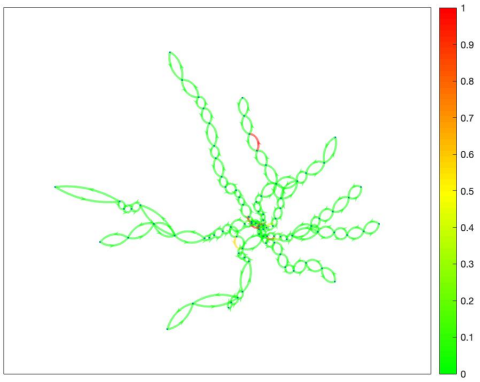
(c) Network graph representing node delays between 2 pm and 2:300 pm for 07-10-17



(d) Network graph representing node delays between 2:30 pm and 3 pm for 07-10-17



(e) Network graph representing transfer delays between 2 pm and 2:30 pm for 07-10-17



(f) Network graph representing transfer delays between 2:30 pm and 3 pm for 07-10-17

Figure A.16: Network graphs representing link, node and transfer delays between 2 pm and 3 pm for 07-10-17

The resulting graphs can be found in Figure A.16. As expected from the stacked histogram, no large link delays are found. Between 2 pm and 2:30 pm, there are only some small delays at a station in the South and a station in the North. In the successive time slice, no significant link delays can be seen. The largest

node delays between 2 pm and 2:30 pm occur at a station in the North, and two stations in the South. In the successive time slice, the largest node delays are found at a station in the North and a station in the South. Furthermore, it can be seen that between 2 pm and 2:30 pm the largest transfer delays can be found at a station in the North and a station near the centre. Between 2:30 pm and 3 pm the largest transfer delays occur at a station in the North.

A.6. Connecting delays in a spatial and temporal way for irregular weekend days

The next step is to find the locations (link, station or transfer station) that are affected by disruptions. This is done by checking for each time slice the locations where delays occur. Again, a treshold of 0.5 minutes is applied. The final output containing the disruptions of the 7^{th} of October, 2017 can be found in Figure A.8. The columns are created the same way as with the week days.

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
1	[14]	[5]	[3]	38	94	0.41	1
2	[14]	[30 32]	[28 30]	146	213	0.69	1
3	[14]	[34 38 39 40]	[32 34 35 36]	301	450	0.67	1
4	[14 15 16 17 18]	[2 5 7 10 11 12 13 14 15 16 17 18 20 22 24 26 28 30 31 32 34 37 38 43 44 45 46 47 49 50 51 52 53 54 56 57 58 63 64 67 68 69 72 73 74 77 82 84 85 88 99 101 102 103 109 114 116 117 118 124 125 127 129 131 133]	[2 3 5 8 9 10 11 12 13 14 15 16 18 20 22 24 26 28 29 30 32 33 34 39 40 41 42 43 44 45 46 47 48 50 51 52 59 60 63 64 65 68 69 70 73 78 80 81 84 92 94 95 96 102 107 109 110 111 117 118 120 122 124 126 128]	15788	73735	0.21	5
5	[14 15 16]	[142 144]	[142 144]	111	1925	0.06	3
6	[14]	[148]	[148]	182	161	1.13	1
7	[15]	[153]	[153]	93	27	3.44	1
8	[15]	[161 163 164]	[161 163 164]	134	177	0.76	1
9	[15]	[176]	[176]	21	244	0.09	1
10	[16]	[40]	[36]	52	247	0.21	1
11	[16]	[110]	[103]	12	164	0.07	1
12	[16]	[170]	[170]	26	263	0.10	1
13	[17]	[84]	[80]	84	423	0.20	1
14	[17]	[88]	[84]	114	159	0.72	1
15	[17]	[91]	[87]	40	445	0.09	1
16	[17 18]	[44 45 46 51 52 56 69 91 97 101 104 106 108 109 110 117 122 129]	[40 41 42 45 46 50 65 87 94 97 99 101 102 103 110 115 122 138]	3244	10255	0.32	2
17	[17]	[136]	[131]	51	133	0.39	1
18	[17]	[148 149]	[148 149]	142	449	0.32	1
19	[17 18]	[151 153]	[151 153]	99	149	0.67	2
20	[17 18]	[161 162]	[161 162]	150	294	0.51	2
21	[17]	[186]	[186]	48	83	0.57	1
22	[18]	[22]	[20]	107	338	0.32	1
23	[18]	[133]	[128]	147	339	0.43	1
24	[18]	[158]	[158]	124	180	0.69	1
25	[18]	[169]	[169]	11	207	0.05	1

Table A.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
26	[19 20 21 22 23 24 25 26 27 28]	[1 2 7 8 10 11 12 13 14 16 17 18 19 20 21 22 23 24 28 30 31 32 34 35 37 38 39 40 43 44 45 46 47 50 51 52 53 56 57 58 63 68 69 70 71 72 73 74 76 77 78 79 80 82 83 84 85 86 88 91 94 95 96 98 99 100 101 102 103 104 105 106 107 108 109 110 112 113 114 120 122 124 129 140 143 144 146 148 149 150 151 152 154 155 163 165 168 169 170 171]	[1 2 5 6 8 9 10 11 12 14 15 16 17 18 19 20 21 22 26 28 29 30 32 33 34 35 36 39 40 41 42 43 44 45 46 47 50 51 52 59 64 65 66 67 68 69 70 72 73 74 75 76 78 79 80 81 82 84 87 90 91 92 93 94 95 96 97 98 99 100 101 102 103 105 106 107 113 115 117 122 133 137 139 140 143 144 146 148 149 150 151 152 154 155 163 165 168 169 170 171]	104016	511900	0.20	10
27	[19]	[172]	[172]	28	736	0.04	1
28	[19]	[192]	[192]	58	48	1.21	1
29	[20]	[1 2 6 7 8 10 12 13 14 15 16 18 19 21 23 24 26 30 31 37 38 40 41 43 44 45 50 51 52 53 56 58 62 64]	[1 2 4 5 6 8 10 11 12 13 14 16 17 19 21 22 24 28 29 33 34 36 37 39 40 41 44 45 46 47 50 52 58 60]	6825	16218	0.42	1
30	[20 21]	[136 137]	[131 132]	399	401	0.99	2
31	[20 21]	[158 160]	[158 160]	43	553	0.08	2
32	[20]	[176]	[176]	106	718	0.15	1
33	[20]	[189]	[189]	11	179	0.06	1
34	[21 22 23 24 25 26 27 28]	[1 2 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 28 29 30 31 32 34 37 38 39 40 41 42 43 44 45 46 47 48 50 51 52 53 54 55 56 57 58 59 60 62 63 64 66 67]	[1 2 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 26 27 28 29 30 32 33 34 35 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 58 59 60 62 63 123]	59279	235752	0.25	8
35	[22]	[2]	[2]	358	751	0.48	1
36	[22]	[174 176]	[174 176]	315	1539	0.20	1
37	[23 24 25]	[114 115]	[107 108]	1189	1822	0.65	3
38	[23]	[119]	[112]	204	144	1.42	1
39	[23]	[126]	[119]	204	806	0.25	1
40	[23]	[133]	[128]	8	485	0.02	1
41	[23]	[136]	[131]	54	104	0.52	1
42	[23]	[188]	[188]	177	95	1.86	1
43	[24]	[94]	[90]	72	891	0.08	1
44	[24]	[160]	[160]	44	125	0.35	1
45	[25]	[73]	[69]	153	599	0.26	1
46	[25]	[133]	[128]	37	388	0.10	1
47	[25]	[136]	[131]	150	119	1.26	1
48	[25]	[141]	[141]	39	395	0.10	1

Table A.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
49	[25]	[178]	[178]	54	215	0.25	1
50	[27 28]	[111 112]	[104 105]	95	376	0.25	2
51	[27]	[141]	[141]	55	541	0.10	1
52	[27 28]	[162 163]	[162 163]	169	323	0.52	2
53	[27 28 29]	[175 178 179]	[175 178 179]	349	2832	0.12	3
54	[28]	[71]	[67]	89	509	0.17	1
55	[28]	[117]	[110]	95	143	0.66	1
56	[28]	[146]	[146]	39	329	0.12	1
57	[28 29]	[169]	[169]	112	1085	0.10	2
58	[29 30 31 32 33]	[1 2 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 29 30 31 32 34 37 38 41 42 43 44 45 46 47 48 50 52 53 54 55 56 57 58 63 64 66 72 73 77 78 79 94 95 97 99 100 101 102 104 105 106 107 108 109 111 112 113 116 130 134 140 143 153 155 157 159]	[1 2 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 24 27 28 29 30 32 33 34 37 38 39 40 41 42 43 44 46 47 48 49 50 51 52 59 60 62 68 69 73 74 75 90 91 92 93 94 95 97 98 99 100 101 102 104 105 106 109 123 125 129 138 140 143 153 155 157 159]	65506	252460	0.26	5
59	[30]	[2]	[2]	542	1268	0.43	1
60	[30]	[88]	[84]	203	223	0.91	1
61	[31 32 33]	[1 2 6 7 8 11 12 13 14 15 16 17 18 19 20 21 22 23 28 30 31 32 34 37 38 39 41 42 43 44 45 46 47 51 53 56 57 58 59 63 64 67 68 73 74 77 78 84 85 88 90]	[1 2 4 5 6 9 10 11 12 13 14 15 16 17 18 19 20 21 26 28 29 30 32 33 34 35 37 38 39 40 41 42 43 45 47 50 51 52 53 59 60 63 64 69 70 73 74 80 81 84 86]	32112	118568	0.27	3
62	[31 32 33 34 35 36 37]	[1 2 4 5 6 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 25 26 28 29 30 31 32 33 34 35 37 38 40 41 42 43 44 45 46 47 50 51 52 53 54 55 56 57 58 59 60 62 63 64 70 71 72 73 74 76 77 78 79 81 82 84 85 87 88 89 94 95 99 100 101 102 103 104 105 111 112 113 115 116 118 120 129 130 132 134 136]	[1 2 3 4 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 22 23 24 26 27 28 29 30 31 32 33 34 36 37 38 39 40 41 42 43 44 45 46 47 48 49 50 51 52 53 54 56 58 59 60 66 67 68 69 70 72 73 74 75 77 78 80 81 83 84 85 90 91 92 93 94 95 96 97 98 104 105 106 108 109 111 113 122 125 127 129 131 133]	100632	442379	0.23	7
63	[31 32]	[120 122]	[113 115]	51	3388	0.02	2
64	[31]	[148]	[148]	101	321	0.32	1
65	[31 32]	[148 150 152]	[148 150 152]	224	1554	0.14	2
66	[31]	[177]	[177]	36	214	0.17	1

Table A.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
67	[32]	[1 2 6 7 8 10 11 12 13 14 15 16 17 22 23 28 32 37 38 39 41 42 43 44 45 46 47 50 51 52 56 58 60 63 64 66]	[1 2 4 5 6 8 9 10 11 12 13 14 15 20 21 26 30 33 34 35 37 38 39 40 41 42 43 44 45 46 50 52 54 59 60 62]	8781	28286	0.31	1
68	[32]	[72 73]	[68 69]	726	1269	0.57	1
69	[32 33 34]	[141]	[141]	44	7781	0.01	3
70	[32 33]	[144 145]	[144 145]	564	1638	0.34	2
71	[32 33]	[188 189]	[188 189]	120	878	0.14	2
72	[33]	[1 2 7 8 11 12 13 14 15 16 17 18 19 20 21 22 23 26 28 32 34 37 38 40 43 44 46 47 50 51 52 53 56 57 58 59 63 64]	[1 2 5 6 9 10 11 12 13 14 15 16 17 18 19 20 21 24 26 30 32 33 34 36 39 40 42 43 44 45 46 47 50 51 52 53 59 60]	10380	30485	0.34	1
73	[33]	[83 84]	[79 80]	228	1353	0.17	1
74	[33]	[128]	[121]	114	939	0.12	1
75	[33]	[160]	[160]	117	148	0.79	1
76	[33 34 35 36 37]	[160 162 163 164]	[160 162 163 164]	737	1865	0.40	5
77	[34 35 36 37]	[151 152 153]	[151 152 153]	1163	1658	0.70	4
78	[34]	[175]	[175]	36	772	0.05	1
79	[35 36]	[144 146 148]	[144 146 148]	220	1443	0.15	2
80	[35]	[182]	[182]	109	213	0.51	1
81	[36]	[189]	[189]	42	208	0.20	1
82	[37]	[136]	[131]	133	106	1.25	1
83	[37]	[170]	[170]	35	615	0.06	1
84	[37]	[193]	[193]	159	206	0.77	1
85	[38 39]	[1 2 6 7 8 10 11 13 14 16 18 19 20 21 22 23 30 31 35 37 38 40 41 43 44 47 50 51 52 56 58 62 63 64 66 77 82 94 95 99 101 102 103 104 105 106 115 119 126 127 129 131 134 136 138]	[1 2 4 5 6 8 9 11 12 14 16 17 18 19 20 21 28 29 33 34 36 37 39 40 43 44 45 46 50 52 58 59 60 62 73 78 90 91 92 94 95 96 97 98 99 108 112 119 120 122 126 129 131 133 135]	18947	54172	0.35	2
86	[38]	[188]	[188]	62	142	0.43	1
87	[39]	[1 2 6 7 8 10 11 12 13 14 16 18 19 20 21 22 23 24 26 28 30 32 37 38 40 41 43 44 45 46 47 50 51 52 53 56 57 58 60 63 64 69 77 79 81 83 90]	[1 2 4 5 6 8 9 10 11 12 14 16 17 18 19 20 21 22 24 26 28 30 33 34 36 37 39 40 41 42 43 44 45 46 47 50 51 52 54 59 60 65 73 75 77 79 86]	8755	20951	0.42	1

Table A.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
88	[39 40 41 42 43 44 45 46 47 48]	[2 5 7 8 10 11 12 13 14 15 16 17 18 19 20 21 22 23 26 28 31 32 34 37 38 39 40 42 43 44 45 46 47 48 49 50 51 52 53 54 55 56 58 59 62 63 64 68 69 71 73 74 75 77 85 86 88 89 94 95 96 97 99 100 102 103 104 105 107 110 111 114 115 116 117 118 127 128 129 130 131 133 135 136 137 140]	[2 3 5 6 8 9 10 11 12 13 14 15 16 17 18 19 20 21 24 26 29 30 32 33 34 35 36 38 39 40 41 42 43 44 45 46 47 48 49 50 52 53 58 59 60 64 65 67 69 70 71 73 81 82 84 85 90 91 92 93 95 96 97 98 100 103 104 107 108 109 110 111 120 121 122 123 124 125 126 128 130 131 132 137 138 140]	53978	280120	0.19	10
89	[39]	[144]	[144]	17	177	0.10	1
90	[39]	[155]	[155]	18	90	0.20	1
91	[39]	[163]	[163]	88	48	1.84	1
92	[39 40]	[175 176 185]	[175 176 185]	180	1841	0.10	2
93	[40]	[2]	[2]	179	770	0.23	1
94	[40]	[133]	[128]	27	156	0.17	1
95	[40]	[169]	[169]	134	595	0.23	1
96	[40]	[184]	[184]	128	90	1.42	1
97	[41 42]	[155 157]	[155 157]	190	430	0.44	2
98	[41]	[160 161 163]	[160 161 163]	494	238	2.08	1
99	[42]	[1 2 4 8 11 13 16 19 20 21 22 23 26 30 32 34 37 38 39 43 44 45 47 50 51 52 56 58 62 64 67 72 74 77 81 96 98 100 101 103 108 114 117 128 129]	[1 2 6 9 11 14 17 18 19 20 21 24 28 30 32 33 34 35 39 40 41 43 44 45 46 50 52 56 58 60 63 68 70 73 77 93 94 96 101 107 110 121 122 137 139]	6173	20666	0.30	1
100	[42 43]	[143 144]	[143 144]	326	1793	0.18	2
101	[42 43 44 45 46 47 48]	[150 152 153 154]	[150 152 153 154]	741	1392	0.53	7
102	[42]	[175]	[175]	16	749	0.02	1
103	[42]	[183]	[183]	210	237	0.89	1
104	[43 44]	[1 2 6 7 8 10 11 12 13 14 15 19 21 23 26 28 37 38 40 43 44 45 47 50 51 54 55 56 58 63 72 77 81 82 85 86 89 95 97 99 100 101 102 103 105 109 110 111 114 126 130 132]	[1 2 4 5 6 8 9 10 11 12 13 17 19 21 24 26 33 34 36 39 40 41 43 44 45 48 49 50 52 59 68 73 77 78 81 82 85 91 92 93 94 95 96 98 102 103 104 107 119 125 127 138]	18344	53580	0.34	2
105	[43]	[163]	[163]	48	42	1.14	1
106	[44]	[1 2 6 8 13 14 16 17 19 23 24 28 30 40 41 42 43 47 48 51 52 53 56 58 62]	[1 2 4 6 11 12 14 15 17 21 22 26 28 36 37 38 39 43 45 46 47 50 52 58 123]	5351	10491	0.51	1

Table A.8 continued from previous page

Disruption ID	Time	Node ID	Link ID	Total passenger delay	Counts	Average delay	Duration
107	[44]	[68 70]	[64 66]	252	1128	0.22	1
108	[44 45 46]	[174 176 179 182 184]	[174 176 179 182 184]	232	1454	0.16	3
109	[46]	[32]	[30]	14	27	0.50	1
110	[46 47 48]	[34 38 39 40]	[32 34 35 36]	719	5145	0.14	3
111	[46]	[58]	[52]	18	57	0.32	1
112	[46]	[62]	[58]	18	316	0.06	1
113	[46 47 48]	[161 162 163 164]	[161 162 163 164]	356	365	0.98	3
114	[46]	[167]	[167]	25	46	0.54	1
115	[47]	[6]	[4]	51	522	0.10	1
116	[47]	[158]	[158]	23	23	0.98	1
117	[48]	[31]	[29]	53	73	0.73	1
118	[48]	[182 184]	[182 184]	58	44	1.31	1

Table A.8: Final output showing disruptions of 7^{th} of October, 2017

A.7. Verification of detected disruptions within irregular weekend days

After the disruptions have been detected, the disruptions within weekend days verified, using the disruption log file of WMATA. A summary of the findings and verification for a selection of irregular weekend days can be found in Table A.9.

	07-10-17	08-10-17	11-02-18	21-04-18	30-06-18
Number of disruptions	118	178	61	72	42
Max total passenger delay	104,016	43,120	105,464	952,737	688,631
Avg total passenger delay	4,538	1,367	6,768	21,148	33,250
Max average passenger delay	3.44	5.35	3.59	5.74	3.55
Avg average passenger delay	0.46	0.79	0.93	1.11	1.04
Max duration (hours)	5	5	15	7.5	9.5
Avg duration (hours)	0.9	0.9	1.3	0.9	1.4
Max counts	511,900	257,593	241,148	1,053,219	1,408,879
Avg counts	18,902	6508	14,922	24,683	68,021
Percentage from log file	25.0	25.0	47.1	54.5	36.0
Percentage in log file	2.7	2.2	3.6	6.3	3.7
Total passenger delay detected	535,429	235,202	412,836	1,718,152	1,396,521
Total passenger delay in log file	215,657	88,583	151,318	1,174,209	1,282,893
Percentage passenger delay in log file	40.3	37.7	36.7	68.3	91.9

Table A.9: Summary disruption detection in weekend days

Table A.9 is calculated the same way as with the week days.

A.7.1. Disruption locations and causes analysis for irregular weekend days

After verifying the detected disruptions of the algorithm using the log file, the locations and causes of disruptions that are found by the algorithm are analysed, and also of disruptions that are not found. This is performed by checking for the five irregular days that are analysed in Section A.7 the locations and causes of occurred disruptions according to the log file, and comparing these to the locations and causes of verified disruptions found by the algorithm.

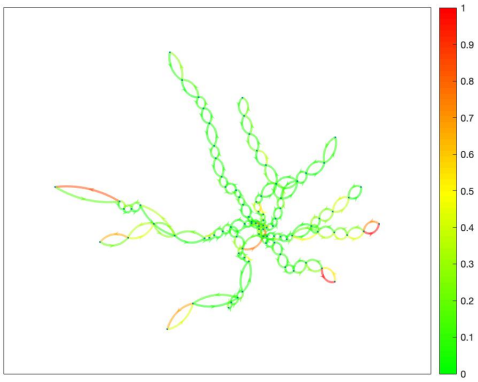
It was found that most of the disruptions that were found by the algorithm were caused by environmental malfunctions (5 times), applied speed restrictions (5 times), door malfunctions (4 times), brake malfunctions

(4 times) and the use of a gap train (4 times). The causes of disruptions that were most often not found by the algorithm are door malfunctions (9 times), propulsion system malfunction (6 times) and using a gap train (4 times). When looking at locations where disruptions were found, the most disruptions that were found by the algorithm occurred at Stadium-Armory (4 times), Farragut North (3 times), Dupont Circle (3 times) and L'Enfant Plaza (3 times). Locations where disruptions occurred that were most often not found by the algorithm are Huntington (4 times), Takoma (3 times), McPherson Square (3 times) and Branch Avenue (3 times).

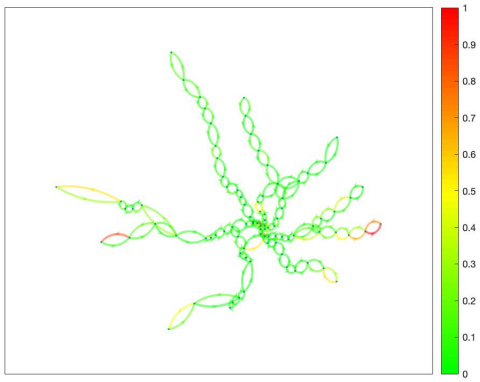
B

Network graphs per cluster: week days

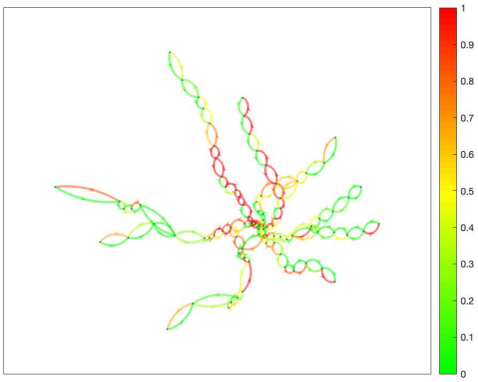
B.1. Network graphs cluster 1: week days



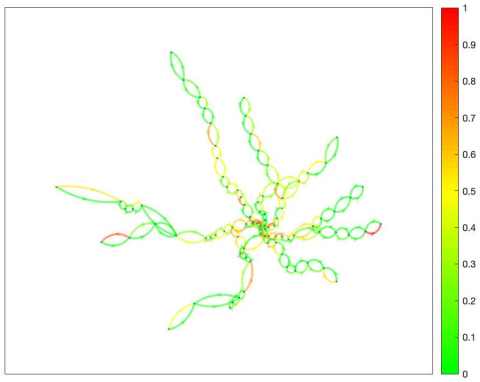
(a) Network graph representing link delays between 9:30 pm and 10 pm for cluster 1



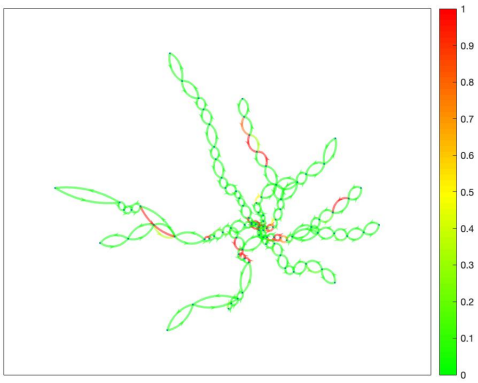
(b) Network graph representing link delays between 10 pm and 10:30 pm for cluster 1



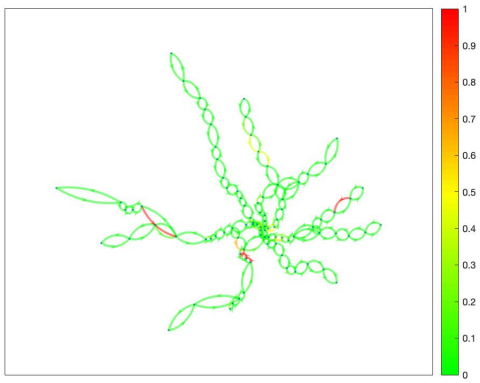
(c) Network graph representing node delays between 9:30 pm and 10 pm for cluster 1



(d) Network graph representing node delays between 10 pm and 10:30 pm for cluster 1



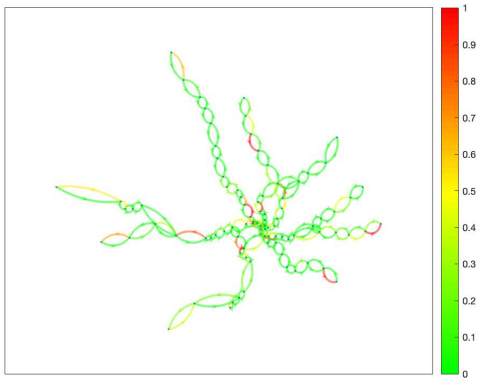
(e) Network graph representing transfer delays between 9:30 pm and 10 pm for cluster 1



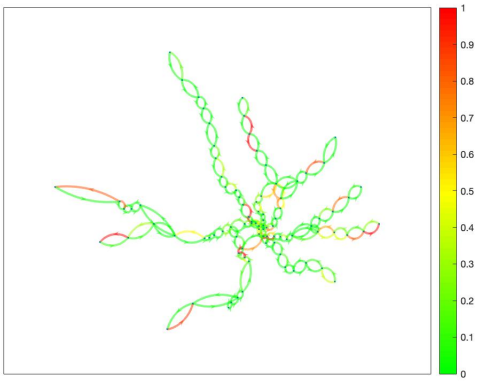
(f) Network graph representing transfer delays between 10 pm and 10:30 pm for cluster 1

Figure B.1: Network graphs representing link, node and transfer delays between 9:30 and 10:30 pm for cluster 1

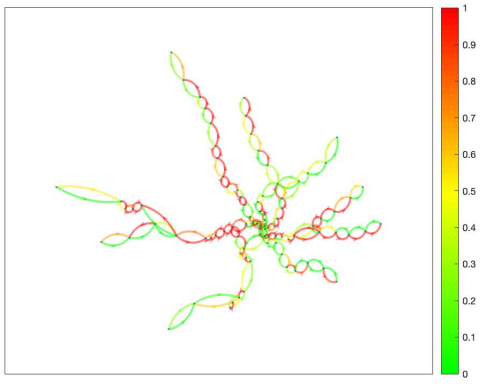
B.2. Network graphs cluster 2: week days



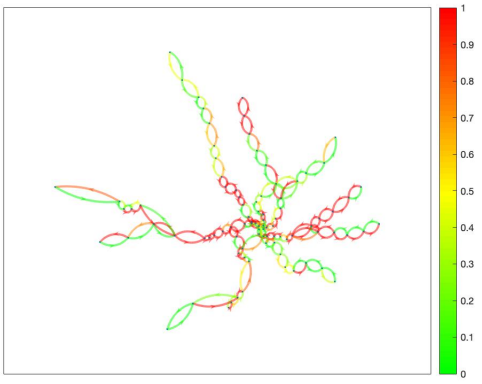
(a) Network graph representing link delays between 5:30 pm and 6 pm for cluster 2



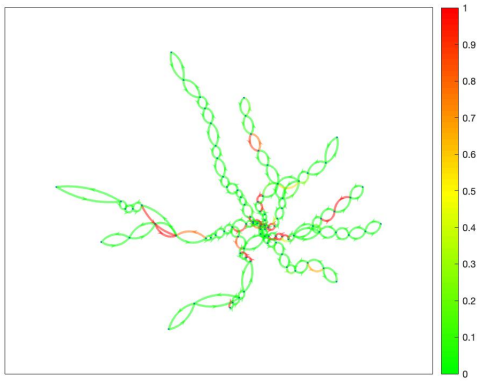
(b) Network graph representing link delays between 6 pm and 6:30 pm for cluster 2



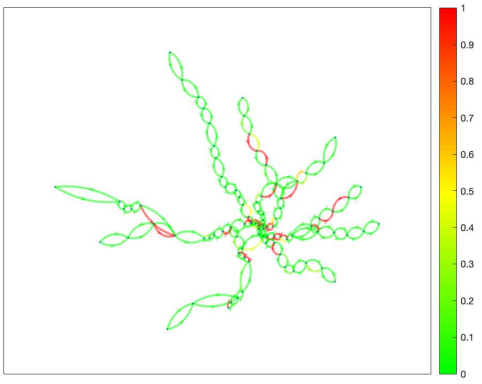
(c) Network graph representing node delays between 5:30 pm and 6 pm for cluster 2



(d) Network graph representing node delays between 6 pm and 6:30 pm for cluster 2



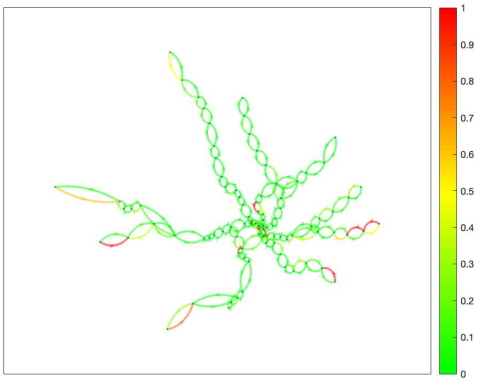
(e) Network graph representing transfer delays between 5:30 pm and 6 pm for cluster 2



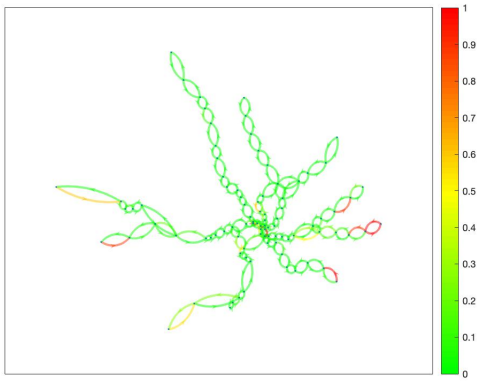
(f) Network graph representing transfer delays between 6 pm and 6:30 pm for cluster 2

Figure B.2: Network graphs representing link, node and transfer delays between 5:30 and 6:30 pm for cluster 2

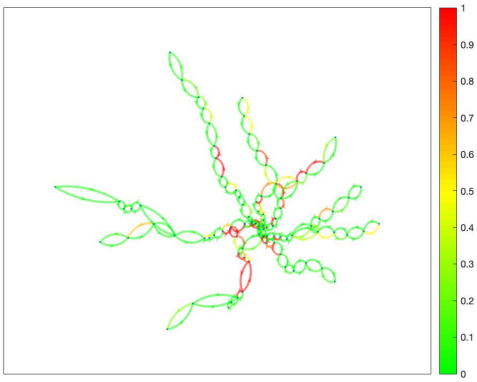
B.3. Network graphs cluster 3: week days



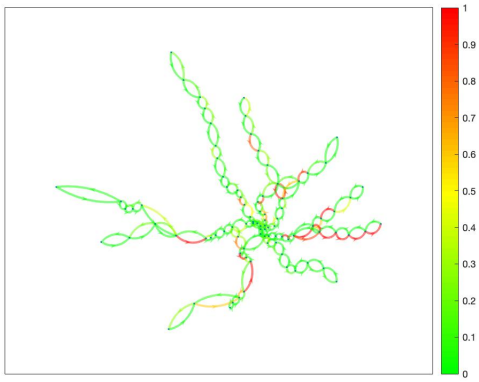
(a) Network graph representing link delays between 5 am and 5:30 am for cluster 3



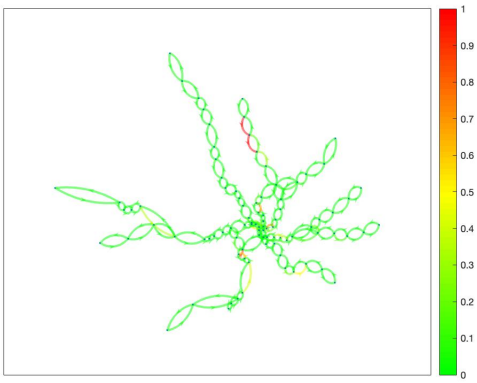
(b) Network graph representing link delays between 5:30 am and 6 am for cluster 3



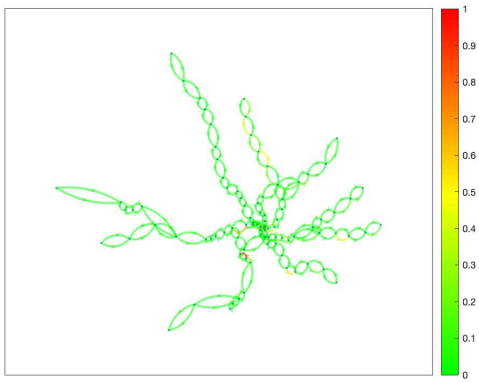
(c) Network graph representing node delays between 5 am and 5:30 am for cluster 3



(d) Network graph representing node delays between 5:30 am and 6 am for cluster 3

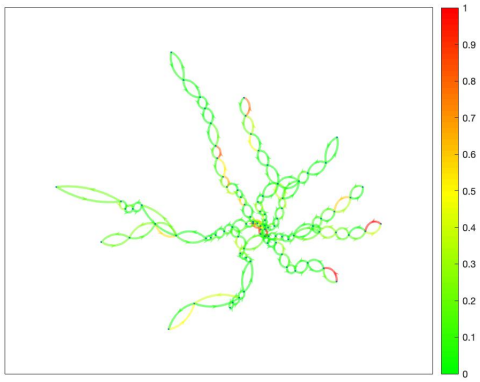


(e) Network graph representing transfer delays between 5 am and 5:30 am for cluster 3

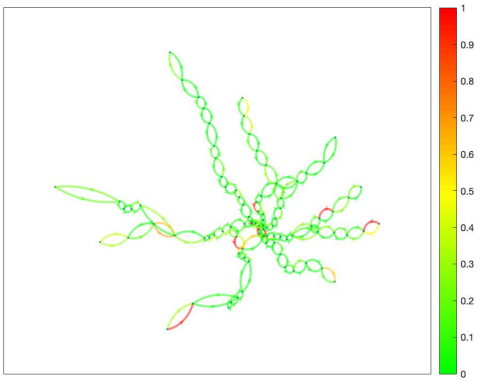


(f) Network graph representing transfer delays between 5:30 am and 6 am for cluster 3

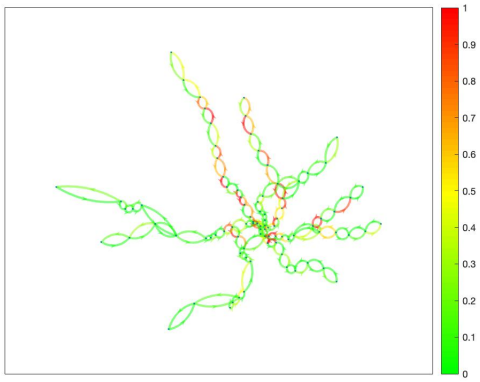
Figure B.3: Network graphs representing link, node and transfer delays between 5 and 6 am for cluster 3



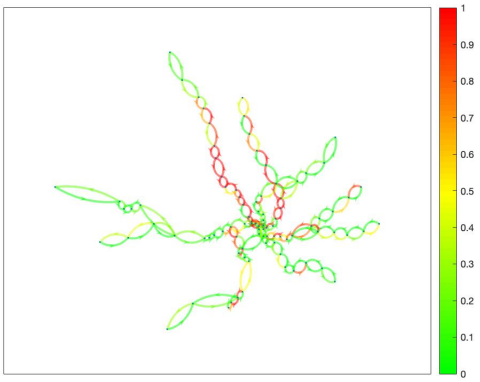
(a) Network graph representing link delays between 8 am and 8:30 am for cluster 3



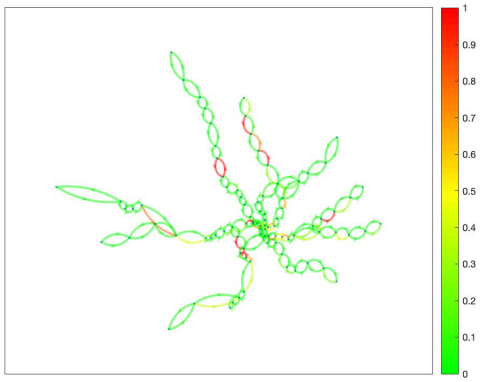
(b) Network graph representing link delays between 8:30 am and 9 am for cluster 3



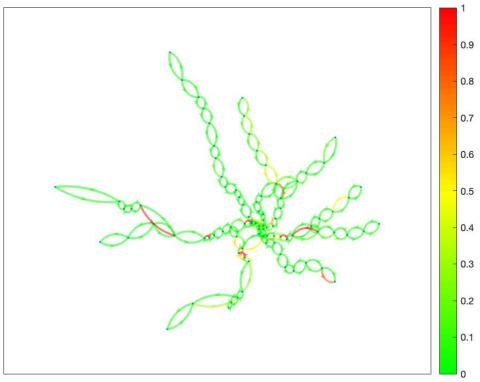
(c) Network graph representing node delays between 8 am and 8:30 am for cluster 3



(d) Network graph representing node delays between 8:30 am and 9 am for cluster 3

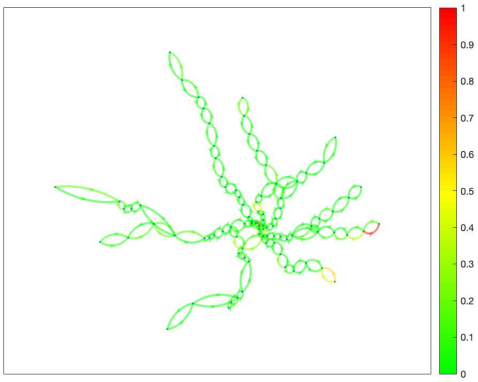


(e) Network graph representing transfer delays between 8 am and 8:30 am for cluster 3

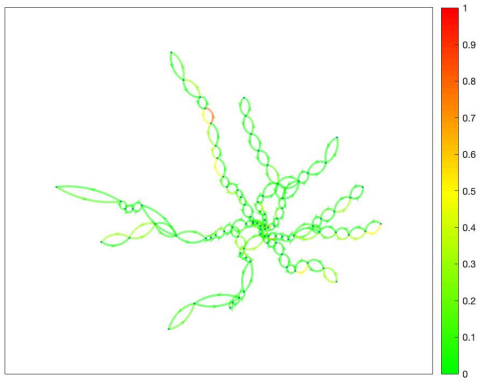


(f) Network graph representing transfer delays between 8:30 am and 9 am for cluster 3

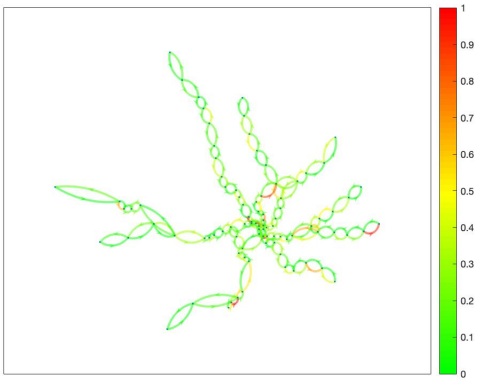
Figure B.4: Network graphs representing link, node and transfer delays between 8 and 9 am for cluster 3



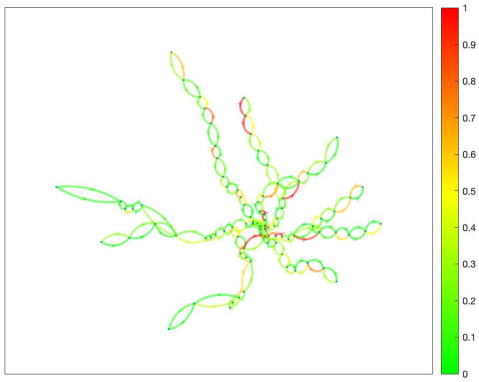
(a) Network graph representing link delays between 4:30 am and 5 pm for cluster 3



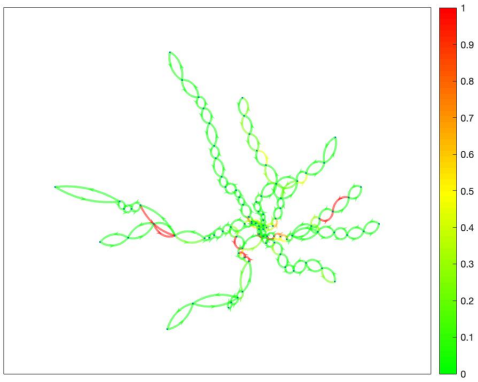
(b) Network graph representing link delays between 5 pm and 5:30 pm for cluster 3



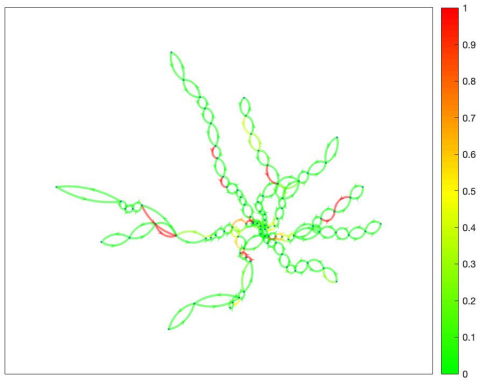
(c) Network graph representing node delays between 4:30 am and 5 pm for cluster 3



(d) Network graph representing node delays between 5 pm and 5:30 pm for cluster 3



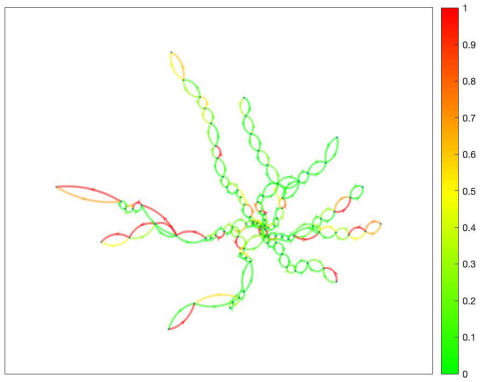
(e) Network graph representing transfer delays between 4:30 am and 5 pm for cluster 3



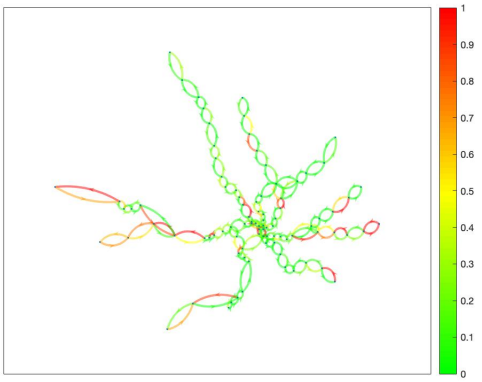
(f) Network graph representing transfer delays between 5 pm and 5:30 pm for cluster 3

Figure B.5: Network graphs representing link, node and transfer delays between 4:30 and 5:30 pm for cluster 3

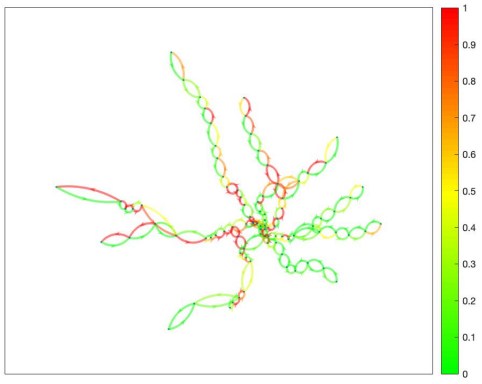
B.4. Network graphs cluster 4: week days



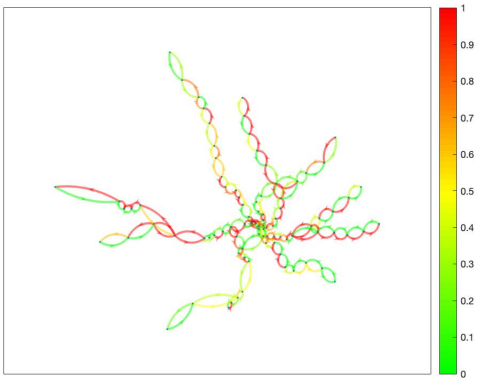
(a) Network graph representing link delays between 7 am and 7:30 am for cluster 4



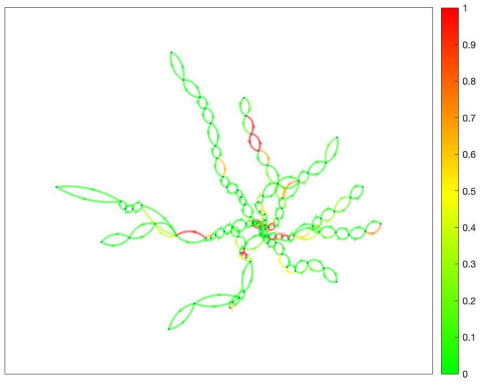
(b) Network graph representing link delays between 7:30 am and 8 am for cluster 4



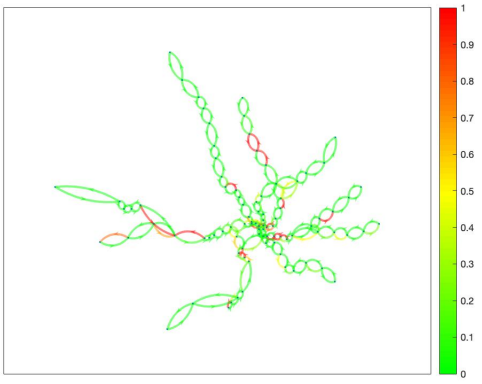
(c) Network graph representing node delays between 7 am and 7:30 am for cluster 4



(d) Network graph representing node delays between 7:30 am and 8 am for cluster 4

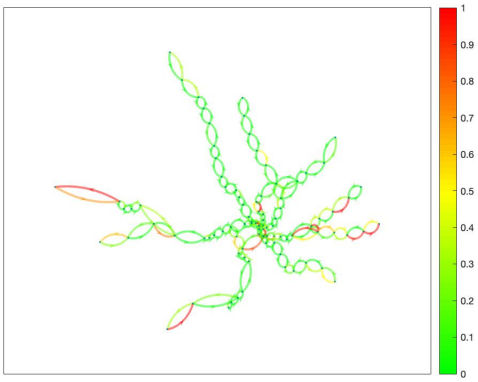


(e) Network graph representing transfer delays between 7 am and 7:30 am for cluster 4

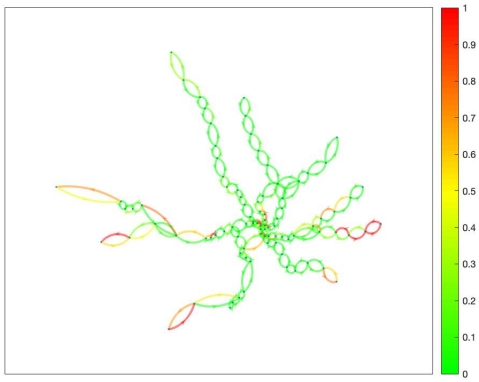


(f) Network graph representing transfer delays between 7:30 am and 8 am for cluster 4

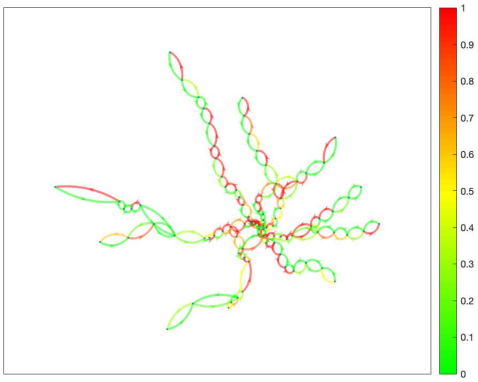
Figure B.6: Network graphs representing link, node and transfer delays between 7 and 8 am for cluster 4



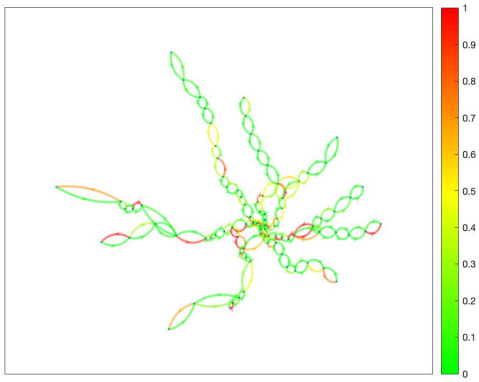
(a) Network graph representing link delays between 9:30 pm and 10 pm for cluster 4



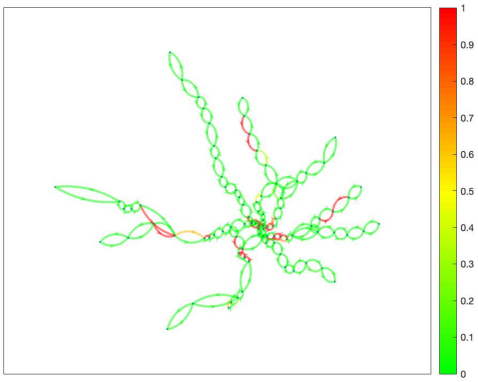
(b) Network graph representing link delays between 10 pm and 10:30 pm for cluster 4



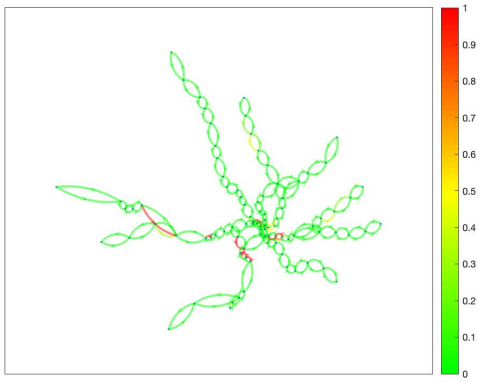
(c) Network graph representing node delays between 9:30 pm and 10 pm for cluster 4



(d) Network graph representing node delays between 10 pm and 10:30 pm for cluster 4



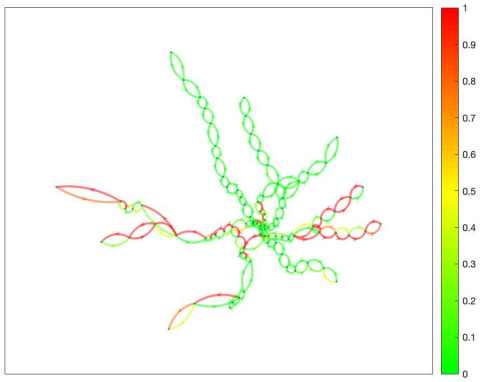
(e) Network graph representing transfer delays between 9:30 pm and 10 pm for cluster 4



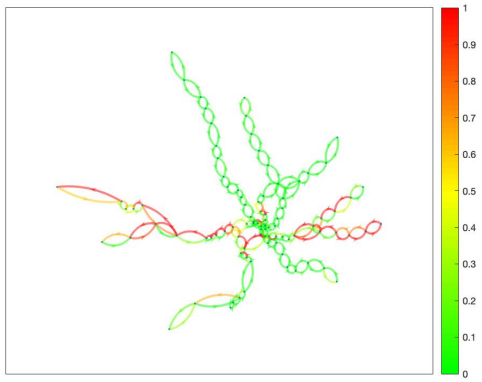
(f) Network graph representing transfer delays between 10 pm and 10:30 pm for cluster 4

Figure B.7: Network graphs representing link, node and transfer delays between 9 and 10 pm for cluster 4

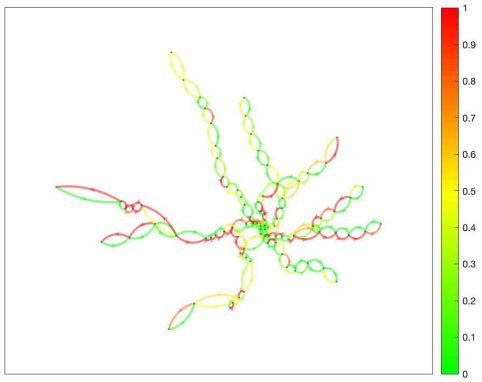
B.5. Network graphs cluster 5: week days



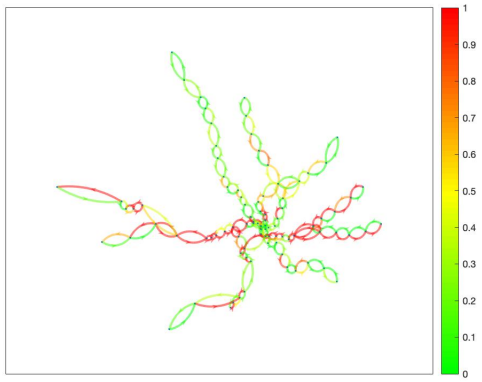
(a) Network graph representing link delays between 5:30 pm and 6 pm for cluster 5



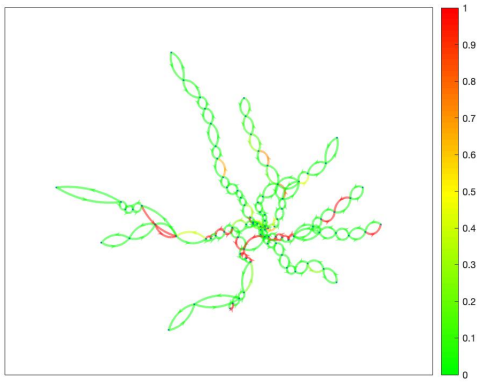
(b) Network graph representing link delays between 6 pm and 6:30 pm for cluster 5



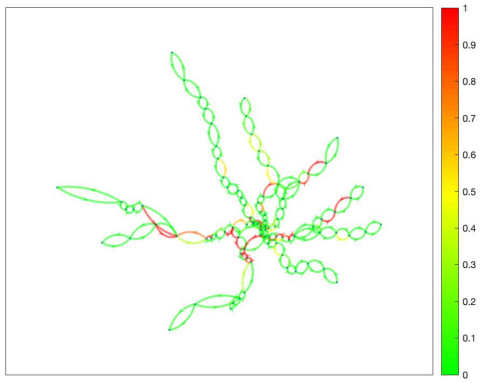
(c) Network graph representing node delays between 5:30 pm and 6 pm for cluster 5



(d) Network graph representing node delays between 6 pm and 6:30 pm for cluster 5



(e) Network graph representing transfer delays between 5:30 pm and 6 pm for cluster 5



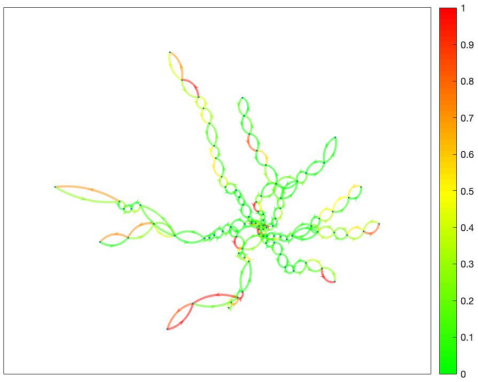
(f) Network graph representing transfer delays between 6 pm and 6:30 pm for cluster 5

Figure B.8: Network graphs representing link, node and transfer delays between 5:30 and 6:30 pm for cluster 5

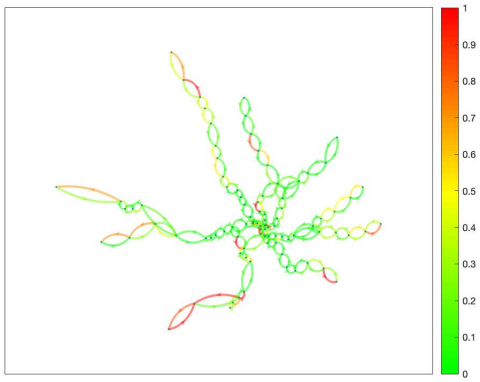
C

Network graphs per cluster: weekend days

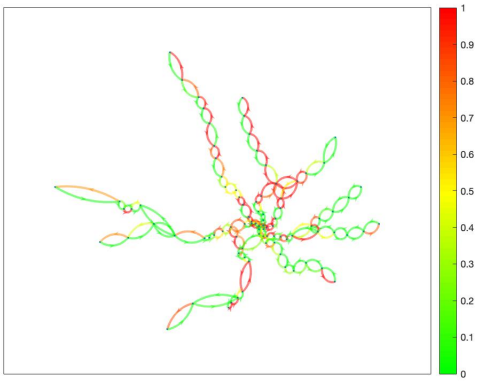
C.1. Network graphs cluster 2: weekend days



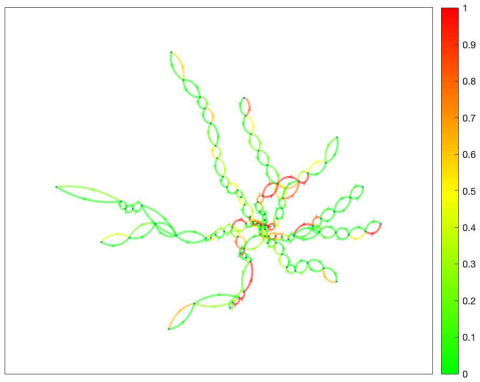
(a) Network graph representing link delays between 9:30 pm and 10 pm for cluster 2



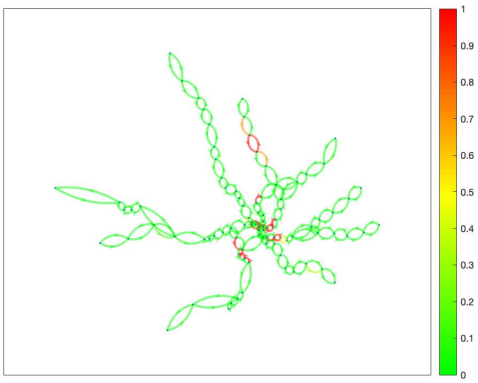
(b) Network graph representing link delays between 10 pm and 10:30 pm for cluster 2



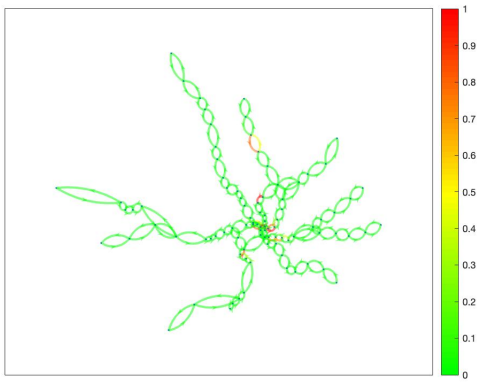
(c) Network graph representing node delays between 9:30 pm and 10 pm for cluster 2



(d) Network graph representing node delays between 10 pm and 10:30 pm for cluster 2



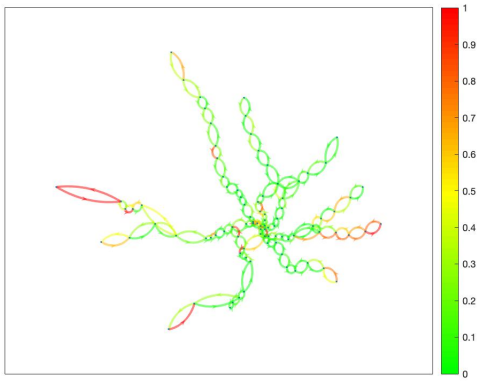
(e) Network graph representing transfer delays between 9:30 pm and 10 pm for cluster 2



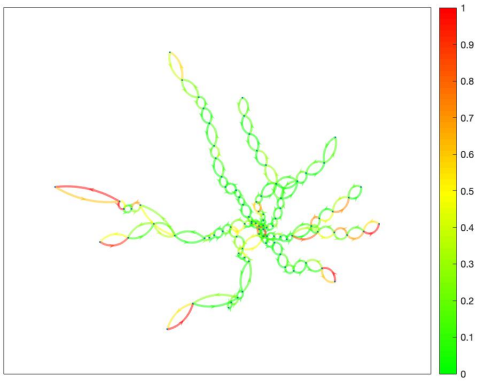
(f) Network graph representing transfer delays between 10 pm and 10:30 pm for cluster 2

Figure C.1: Network graphs representing link, node and transfer delays between 9:30 and 10:30 pm for cluster 2

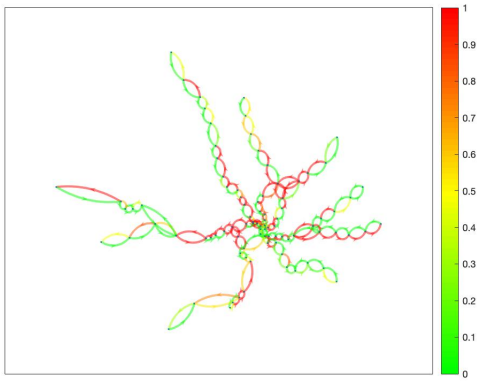
C.2. Network graphs cluster 3: weekend days



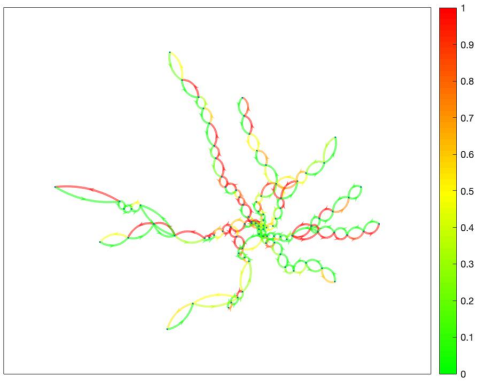
(a) Network graph representing link delays between 8 am and 8:30 am for cluster 3



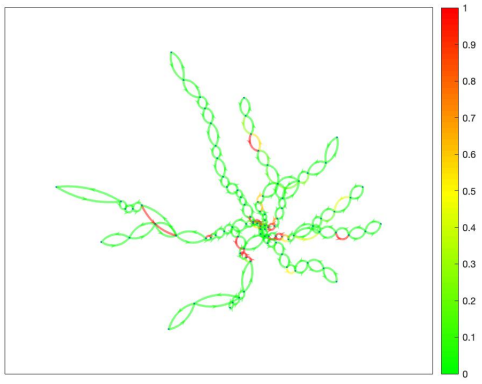
(b) Network graph representing link delays between 8:30 am and 9 am for cluster 3



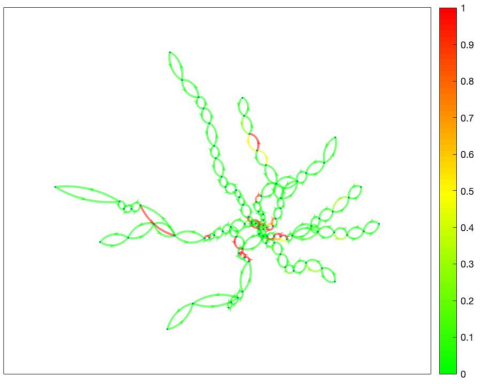
(c) Network graph representing node delays between 8 am and 8:30 am for cluster 3



(d) Network graph representing node delays between 8:30 am and 9 am for cluster 3



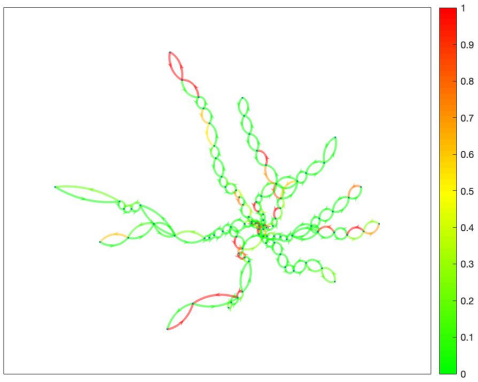
(e) Network graph representing transfer delays between 8 am and 8:30 am for cluster 3



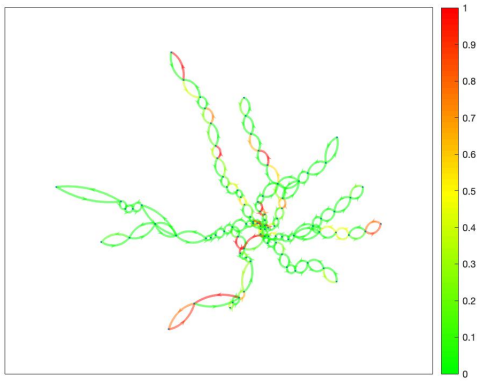
(f) Network graph representing transfer delays between 8:30 am and 9 am for cluster 3

Figure C.2: Network graphs representing link, node and transfer delays between 8 am and 9 am for cluster 3

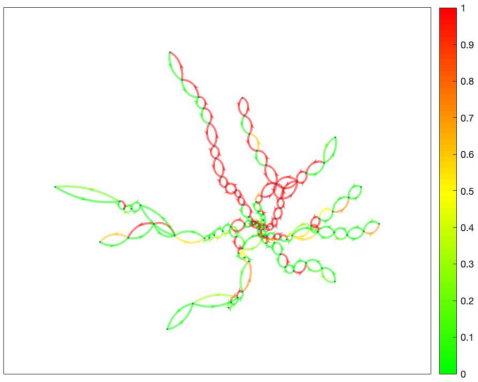
C.3. Network graphs cluster 4: weekend days



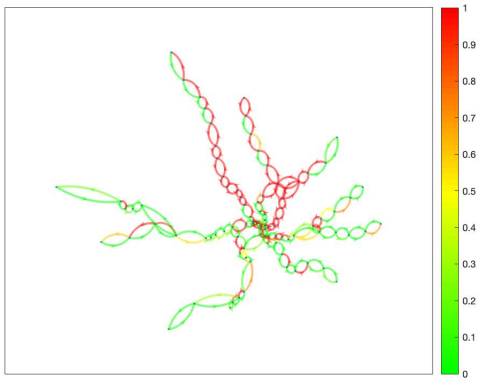
(a) Network graph representing link delays between 1:30 pm and 2 pm for cluster 4



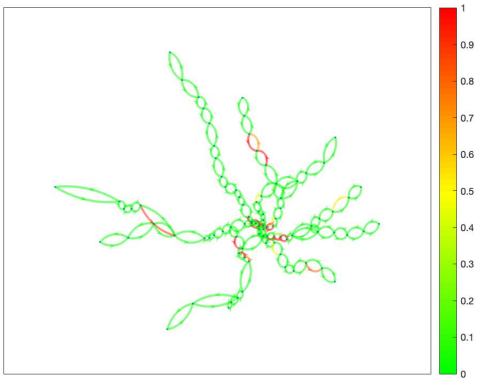
(b) Network graph representing link delays between 2 pm and 2:30 pm for cluster 4



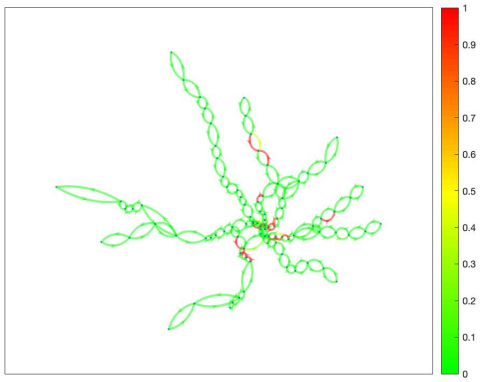
(c) Network graph representing node delays between 1:30 pm and 2 pm for cluster 4



(d) Network graph representing node delays between 2 pm and 2:30 pm for cluster 4



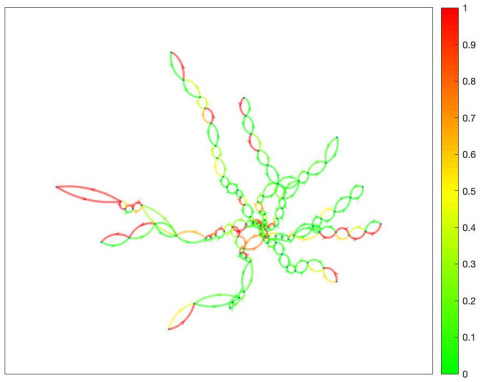
(e) Network graph representing transfer delays between 1:30 pm and 2 pm for cluster 4



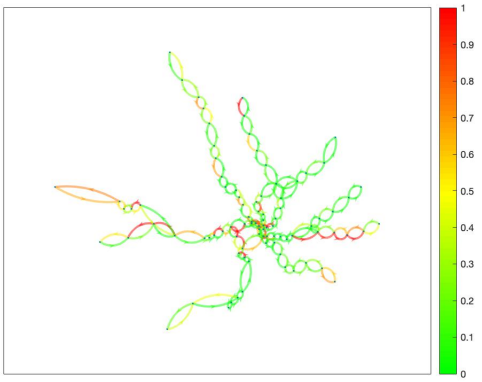
(f) Network graph representing transfer delays between 2 pm and 2:30 pm for cluster 4

Figure C.3: Network graphs representing link, node and transfer delays between 1:30 and 2:30 pm for cluster 4

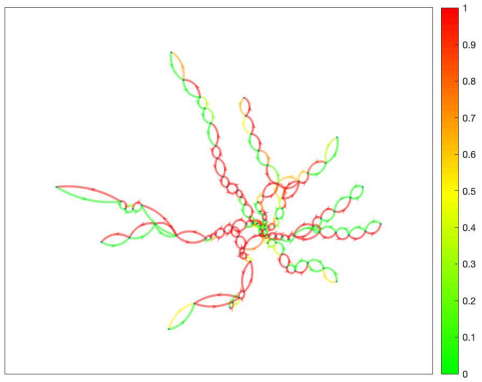
C.4. Network graphs cluster 5: weekend days



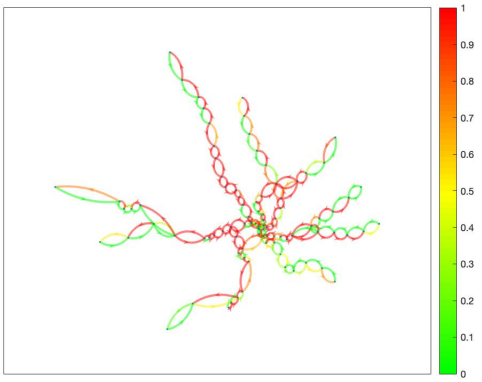
(a) Network graph representing link delays between 11:30 am and noon for cluster 5



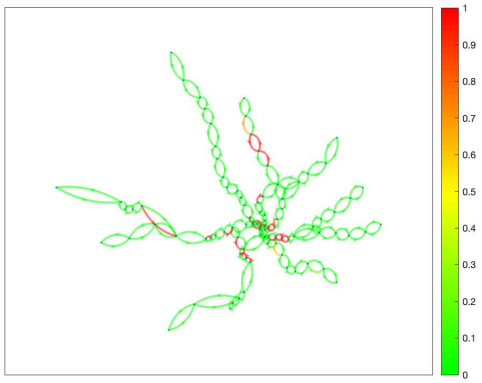
(b) Network graph representing link delays between noon and 12:30 pm for cluster 5



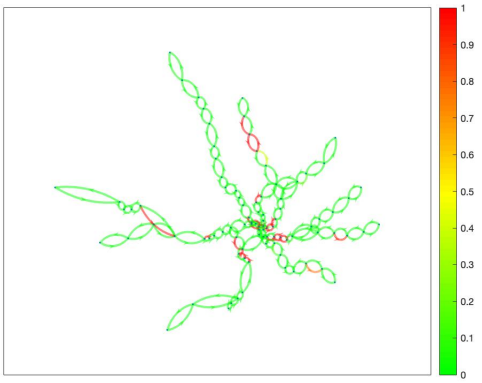
(c) Network graph representing node delays between 11:30 am and noon for cluster 5



(d) Network graph representing node delays between noon and 12:30 pm for cluster 5



(e) Network graph representing transfer delays between 11:30 am and noon for cluster 5



(f) Network graph representing transfer delays between noon and 12:30 pm for cluster 5

Figure C.4: Network graphs representing link, node and transfer delays between 11:30 am and 12:30 pm for cluster 5

D

Station names

Station number	Station name	Station number	Station name	Station number	Station name
1	Metro Center	66	Foggy Bottom	131	West Hyattsville
2	Metro Center	67	Foggy Bottom	132	West Hyattsville
3	Metro Center	68	Rosslyn	133	Prince Georges Plza
4	Metro Center	69	Rosslyn	134	Prince Georges Plza
5	Farragut North	70	Rosslyn	135	College Park - UM
6	Farragut North	71	Arlington Cemetery	136	College Park - UM
7	Farragut North	72	Arlington Cemetery	137	Greenbelt
8	Dupont Circle	73	Pentagon	138	Archives
9	Dupont Circle	74	Pentagon	139	Archives
10	Woodley Park	75	Pentagon	140	Waterfront
11	Woodley Park	76	Pentagon City	141	Waterfront
12	Cleveland Park	77	Pentagon City	142	Navy Yard
13	Cleveland Park	78	Crystal City	143	Navy Yard
14	Van Ness - UDC	79	Crystal City	144	Annacostia
15	Van Ness - UDC	80	National Airport	145	Annacostia
16	Tenleytown - AU	81	National Airport	146	Congress Heights
17	Tenleytown - AU	82	Braddock Road	147	Congress Heights
18	Friendship Heights	83	Braddock Road	148	Southern Avenue
19	Friendship Heights	84	King Street	149	Southern Avenue
20	Bethesda	85	King Street	150	Naylor Road
21	Bethesda	86	King Street	151	Naylor Road
22	Medical Center	87	Eisenhower Ave.	152	Suitland
23	Medical Center	88	Eisenhower Ave.	153	Suitland
24	Grosvenor	89	Huntington	154	Branch Avenue
25	Grosvenor	90	Federal Triangle	155	Benning Road
26	White Flint	91	Federal Triangle	156	Benning Road
27	White Flint	92	Smithsonian	157	Benning Road
28	Twinbrook	93	Smithsonian	158	Capitol Heights
29	Twinbrook	94	L'Enfant Plaza	159	Capitol Heights
30	Rockville	95	L'Enfant Plaza	160	Addison Road
31	Rockville	96	L'Enfant Plaza	161	Addison Road
32	Shady Grove	97	L'Enfant Plaza	162	Morgan Boulevard
33	Gallery Place	98	L'Enfant Plaza	163	Morgan Boulevard
34	Gallery Place	99	Federal Center SW	164	Largo Town Center
35	Gallery Place	100	Federal Center SW	165	Van Dorn
36	Gallery Place	101	Capitol South	166	Van Dorn
37	Judiciary Square	102	Capitol South	167	Franconia-Sprgfield
38	Judiciary Square	103	Eastern Market	168	Court House
39	Union Station	104	Eastern Market	169	Court House
40	Union Station	105	Potomac Avenue	170	Clarendon
41	Rhode Island Ave	106	Potomac Avenue	171	Clarendon
42	Rhode Island Ave	107	Stad-Armory	172	Virginia Square
43	Brookland - CUA	108	Stad-Armory	173	Virginia Square
44	Brookland - CUA	109	Stad-Armory	174	Ballston
45	Brookland - CUA	110	Minnesota Avenue	175	Ballston
46	Fort Totten	111	Minnesota Avenue	176	E. Falls Church
47	Fort Totten	112	Minnesota Avenue	177	E. Falls Church
48	Fort Totten	113	Deanwood	178	E. Falls Church
49	Fort Totten	114	Deanwood	179	West Falls Church
50	Takoma	115	Cheverly	180	West Falls Church
51	Takoma	116	Cheverly	181	West Falls Church
52	Silver Spring	117	Landover	182	Dunn Loring
53	Silver Spring	118	Landover	183	Dunn Loring
54	Forest Glen	119	New Carrollton	184	Vienna
55	Forest Glen	120	Mt Vern UDC	185	McLean
56	Wheaton	121	Mt Vern UDC	186	McLean
57	Wheaton	122	Shaw - Howard U	187	Tysons Corner
58	Glenmont	123	Shaw - Howard U	188	Tysons Corner
59	New York Avenue	124	U Street - Cardozo	189	Greensboro
60	New York Avenue	125	U Street - Cardozo	190	Greensboro
61	McPherson Square	126	Columbia Heights	191	Spring Hill
62	McPherson Square	127	Columbia Heights	192	Spring Hill
63	McPherson Square	128	Georgia - Petworth	193	Wiehle Avenue
64	Farragut West	129	Georgia - Petworth		
65	Farragut West	130	West Hyattsville		

Figure D.1: Station numbers and their corresponding names

E

Paper

AUTOMATED OFFLINE DETECTION OF DISRUPTIONS USING SMART CARD DATA

Faye Jasperse
Department of Transport and Planning
Delft University of Technology
Delft, the Netherlands
f.z.g.jasperse@student.tudelft.nl

January 16, 2020

ABSTRACT

Service reliability is one of the most important performance measures to public transport users. Detecting disruptions helps to measure service reliability, which can be used by public transport operators to improve this reliability. In this paper, a methodology is described to automatically detect disruptions offline, using smart card data. The day-to-day regularity of delays is investigated using hierarchical clustering on a training set, to distinguish between regular and irregular delays. The clustering result is used to create a probabilistic classifier. This classifier is applied to the test set to find days that do not correspond to a regular pattern: irregular days. After that, disruptions are detected within the irregular days. The outcomes of this study can be applied in multiple ways. Locations where disruptions have occurred can be found and the related passenger delay can be calculated. This can help public transport operators to prioritise which locations to focus on to reduce passenger delays. Furthermore, not only public transport networks, but also other networks can benefit from the outcomes of this study. Speed data of road networks could be used to find disruptions that are caused by accidents, instead of regular traffic jams. On top of that, this study could be used as a step towards real-time disruption detection, for both public transport and road networks.

Keywords Public transport · Offline disruption detection · Smart card data · Hierarchical clustering · Probabilistic classification · Machine learning

1 Introduction

Service reliability is one of the most important performance measures to public transport users. It affects both their perception of service quality as well as their travel behaviour. Public transport operators can only understand and improve the reliability if they are able to measure it. Automated disruption detection can help public transport operators to monitor their service performance, therefore this can aid in improving the reliability and reducing passenger delays. It provides them with the duration of disruptions and which locations are most often affected, and therefore supports them to prioritise what locations to focus on to reduce passenger delays. This is also of financial importance: many operators currently offer delay compensations. The aim of this study is to automatically detect disruptions offline in a public transport system, using smart card data. Past research on disruption detection in public transport systems includes detection based on twitter data (Ji et al., 2018) and clustering demand to detect disruptions (Briand et al., 2018). Twitter data was used to investigate if disruptions cause travel delays for passengers (Ji et al., 2018). Clustering was used to identify disruptions in public transport, considering changes in transport demand (Briand et al., 2018). A disruption was defined as an unexpected increase or decrease of tap-in and tap-out data, compared to registrations under regular conditions.

Smart card data is an interesting source to potentially detect disruptions, as it makes it possible to quantify delays on a passenger level. To the best of our knowledge, automated offline detection of disruptions in metro networks using

a data-driven approach with smart card data has not been utilised so far for detecting disruptions in public transport systems. To this end, a new data-driven method is proposed in this study to automatically detect disruptions offline.

This paper is organised as follows: after describing the methodology in the next section, the Washington Metropolitan Area Transit Authority (WMATA) network is used as a case study to demonstrate the applicability of the proposed approach and results are reported. The average passenger delays for each station, track and transfer station that are calculated for the period September 2017 until August 2018, using smart card data of the WMATA network, are used as an input (Krishnakumari et al., 2019). This paper ends with the conclusion.

2 Methodology

First, smart card data is used to calculate the average passenger delay at each station, track segment and interchange station in the network. In the context of this study, a disruption is defined as a delay that does not occur regularly in the network. To distinguish between a delay that occurs regularly and a disruption, the day-to-day regularity of delays must be investigated. When delays fit a similar pattern, for example when a delay regularly occurs at the morning peak, the delays belonging to this pattern are not considered to be a disruption. To investigate the day-to-day regularity, days can be clustered. As can be seen in Figure 1, splitting the data set into training data and test data constitutes the first step in our methodology framework. The training data is used to perform the clustering in the second step. This second step constitutes of distinguishing between regular and irregular days. The output of this step are days that have irregular delay patterns, these are the days that potentially contain disruptions. The third step is to detect the disruptions within these irregular days. Finally, we identify areas in the network with dissimilar delay patterns, the disruptions.

2.1 Hierarchical clustering

For the second step, delays are clustered per day to get insight into regular and irregular days. One part of the data set is used as a training set. Hierarchical clustering is applied to the days belonging to this training set. Hierarchical clustering is chosen because it does not require the number of clusters beforehand. Instead, the cluster merging can be performed using a dendrogram and it does not require a pre-selection of initial centroids.

Hierarchical clustering is applied with three different specifications: first based on the Euclidean distance between the delay vectors of the days, second based on the structural similarity index (SSIM) of the days, and finally these two are combined. The SSIM index is used to measure the similarity between two images (Wang et al., 2004). As the delay for all network elements can be represented as a vector, this method can be applied to cluster days based on their SSIM index. The SSIM index consists of a distance metric that compares the mean values of two vectors, a metric to compare the standard deviation of two values and a metric to compare the structure of the vectors.

After the days are clustered based on both the Euclidean distance as well as the SSIM index, the resulting cluster IDs are used to create new vectors for each day. Consequently, each day is represented by a vector with a length of two: the cluster ID resulting from the clustering based on the Euclidean distance and the cluster ID resulting from the clustering based on the SSIM index. Finally, hierarchical clustering is applied for the third time to these combined vectors. This results in a number of classes that each contains a certain number of days.

There is a probability that irregular days are clustered together, therefore the centroid of each class is analysed to see if the class consists of regular or irregular days. To this end, histograms are created of the stacked link, node and transfer delays for each time slice and by generating a graph of the network displaying the respective delays. When the histogram of a certain centroid displays significantly larger delays than the other centroids and the graph of the network shows large delays, the cluster represents irregular days.

2.2 Probabilistic classification

To determine for days from the testing data whether they belong to the irregular days or regular days, the clustering result from the training data is used to create a probabilistic classifier. This classifier is then applied to the delay data that has not been used as a training set, i.e. the test set. For each day in the test set, this results in the membership probability per cluster. When the probability is smaller than a certain p for a given day and cluster combination, or when a day has the highest probability to belong to a cluster that represents irregular days, the day is considered to be an irregular day.

2.3 Detecting disruptions within irregular days

After distinguishing between regular and irregular days, the third step is executed: detecting disruptions within the irregular days. The differences between the delays of the irregular day and the centroid of the cluster that it has the highest probability to belong to, are considered to be irregular delays. Multiple delays can belong to the same disruption, as delays can spread over the network. Therefore, the delays are analysed to see if they are associated with network elements that are connected. This is not only done in terms of spatial proximity, but also in terms of temporal proximity, as delays can propagate over time. This results in areas in the network that contain delays (disruptions). The average passenger delay is defined as the delay that is experienced by a passenger while travelling over a certain link or node. The total passenger delay is the total delay incurred by all the passengers that traverse a certain link or node. This finally leads to an output containing for each disruption the duration of the disruption, the affected stations and links, the average passenger and total passenger delay.

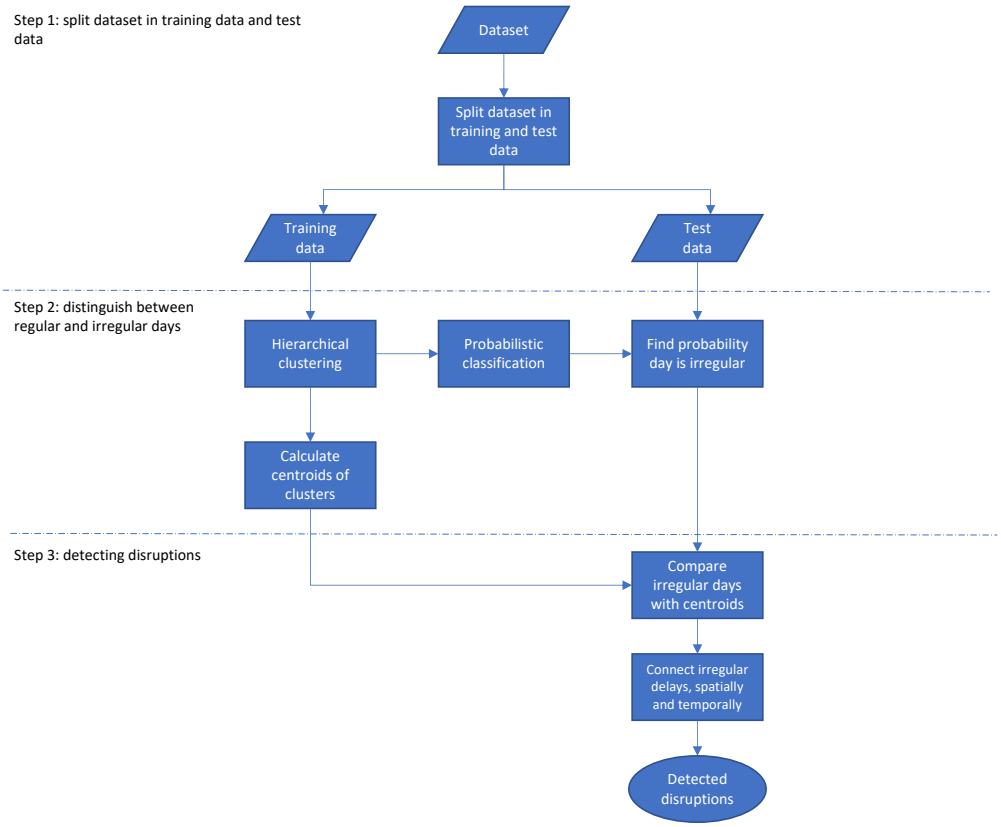


Figure 1: Methodology framework

3 Case study

In this research, the metro network of WMATA is used as a case study. This metro network in the metropolitan of Washington is the third largest heavy rail transit system with a track length of 118 miles (190 kilometres), 6 rail lines, 91 stations and 1144 railcars. In 2018, 174 million trips were made with WMATA (WMATA, 2019b). An overview of the metro network can be found in Figure 2.

3.1 Available data

The data that are used in this study have been made available by WMATA. These data consist of smart card data containing the tap-in and tap-out time stamps and a disruption log file.

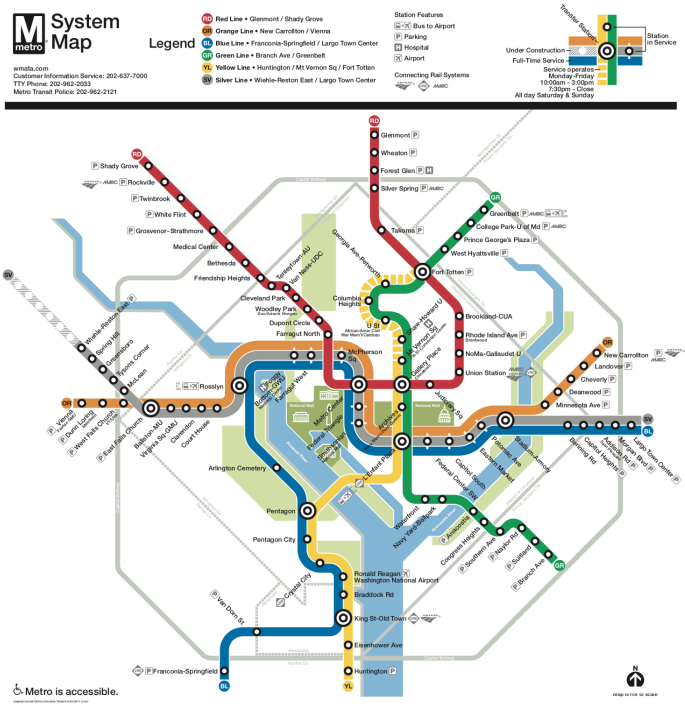


Figure 2: WMATA metro network (WMATA, 2019c)

3.1.1 Smart card data

The smart card data that are used to calculate the average passenger delays consist of all tap-in and tap-out time stamps of passengers travelling between August 2017 and August 2018. As a smart card is required to use the metro network, no trips are missing from these data. Furthermore, the passenger counts have been calculated, using these smart card data as an input. The smart card data have been analysed to calculate the average delay per passenger for each node, link and transfer station.

3.1.2 Disruption log file

The disruption log file contains all disruptions occurring between August 2017 and August 2018. This file has been used to verify the disruptions detected by the algorithm that has been developed in this research.

3.2 Input data used for this study

The average passenger delays calculated by Krishnakumari et al. (2019) will be used as an input for this study. The WMATA network consists of 95 stations. 193 different links can be distinguished when assigning a different number to each link with a different direction. To be able to incorporate direction within the nodes, the nodes are renumbered. One station can get assigned multiple numbers, depending on the link it is connected to. Therefore, also 193 station numbers are distinguished. This is also the case for transfer stations; when a station is not a transfer station, the average passenger transfer delay values at that station will be 0. As the average passenger delay for each track element has been calculated for every half an hour, each day consists of 48 time slices. The data for each day can then be represented as one long vector, with the average passenger node, link and transfer delay for each time slice. Each vector has thus a length of (193*3*48).

4 Results

The average passenger delays have been calculated using the smart card data made available by WMATA (Krishnakumari et al., 2019). The node, link and transfer delays are calculated over an aggregated time slice of half an hour, hence there are 48 time slices per day available as input for this study for the period of September 2017 until August 2018. The days are represented as a vector of node, link and transfer delays for all time slices. It is chosen to use the days of

weeks with an odd number as a training set, and the days of weeks with an even number as a test set. After the first clustering of the days of the odd weeks, we find that most of the clusters contain either mostly weekdays or mostly weekend days. It was therefore decided to separate the clustering for week and weekend days. The resulting truncated dendrogram for the combined clustering of the weekdays of the odd weeks can be found in Figure 3. This dendrogram is the result of clustering the vectors consisting of the cluster IDs based on both the clustering using Euclidean distances as well as the SSIM index. Truncation is used to condense the dendrogram and only display the cluster sizes instead of all sample indices separately. The truncated dendrogram shows only the last p merged clusters, where p is the number of clusters, resulting from the location where the dendrogram is cut.

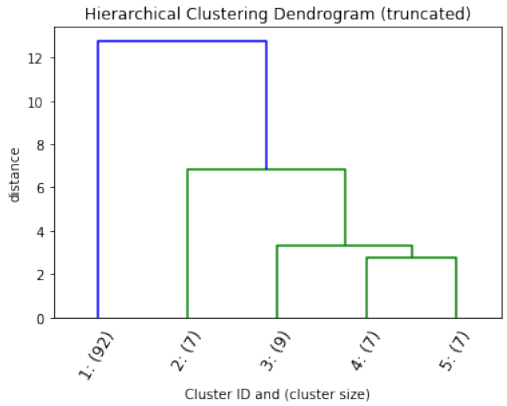
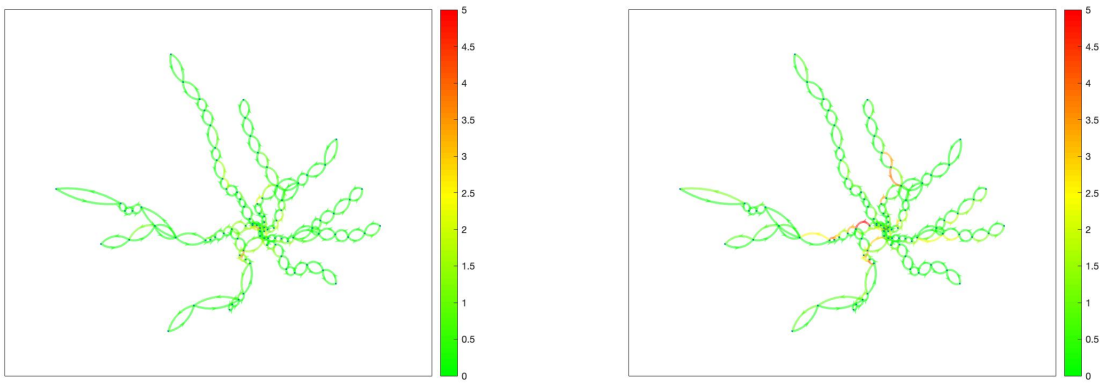


Figure 3: Truncated dendrogram of combined clustering for weekdays

Cluster 1 is by far the largest cluster, so it is expected that this cluster consists of regular days. To investigate if a cluster consists of regular or irregular days, the centroids of each cluster are inspected. This is done by creating a histogram of the stacked link, node and transfer delays for each time slice and by generating a graph of the network displaying the respective delays, as shown in Figure 4. The morning peak in the WMATA network on Monday to Friday is from 5 a.m. until 9:30 a.m. (WMATA, 2019a). Based on the histograms for each centroid, we conclude that the largest average passenger delays of the morning peak in the WMATA metro network occur around 5:30 a.m. When comparing the node delays of centroid 1 and 2 in the network graph at 5:30 a.m. (see Figure 4), it can be seen that centroid 2, which includes fewer days exhibits much larger delays. Cluster 3, 4, and 5 are analysed in the same way and it was found that clusters 2, 3, 4 and 5 all contain larger delays and are on top of that much smaller clusters, so these clusters represent the irregular days. Cluster 1 represents the regular days.



(a) Network graph representing node delays between 5 am and 5:30 am for cluster 1

(b) Network graph representing node delays between 5 am and 5:30 am for cluster 2

Figure 4: Network graphs representing node delays between 5 am and 5:30 am for cluster 1 and 2

Because cluster 2, 3, 4 and 5 represent irregular days, days that have the highest probability to belong to one of these clusters are considered to be an irregular day with high likelihood. Days that do not resemble any of the clusters with a probability of belonging to each one of the clusters lower than 50%, are also considered to be irregular. When using these conditions when applying probabilistic classification, 9 irregular days are found in the weekdays of weeks included in the testing data set, i.e. weeks with an even number. In Figure 5, the difference between the first irregular day, the 4^th of September 2018, and the regular centroid it has the highest probability to belong to, centroid 1, is shown as a stacked histogram. It can be observed that after 7:30 am, larger delays occur than regular.

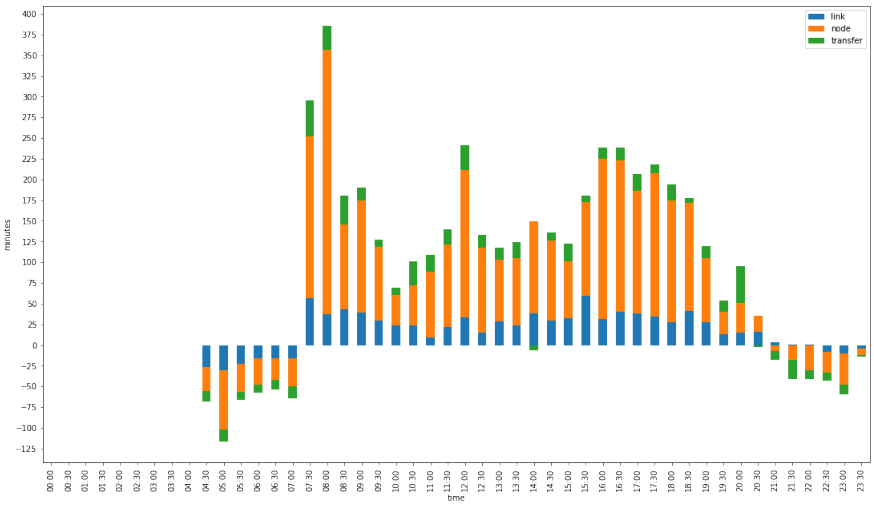


Figure 5: Stacked histogram of link, node and transfer delays of 04-09-17

The next step is to connect the locations where delays occur in a spatial and temporal way, to find the disruptions. This way, 43 disruptions were found on the 4^th of September, with an average duration of 1.4 hours. These disruptions have been validated, using the disruption log file of WMATA. For the 4^th of September, 42.9% of the disruptions in the log file have been found. A possible reason why not all disruptions from the log file have been detected, is that not all disruptions cause delays and vice versa. However, when calculating the percentage of the total passenger delay that has been verified using the log file, for the 4^th of September 90.3% of the passenger delay is verified. The same analysis has been executed for the weekend days.

5 Conclusion

Disruptions in the WMATA network have been detected automatically offline, using smart card data as an input. The methods used are hierarchical clustering, machine learning using probabilistic classification, and algorithms to connect disruptions in both a spatial and temporal way. Interestingly, more irregular days were found in weekend days than in week days, while the number of weekend days was naturally smaller. This might be due to the fact that track works are usually planned during weekends, and events are more often organised during weekends. Therefore, weekends might follow a less regular pattern. Furthermore, for the weekend day clusters, there were no clear morning and afternoon peaks, while these could be observed for most of the week day clusters. Another interesting fact is that, within the irregular week days, 3 of the 9 days were a holiday, while 5 other days were during the summer holiday and suffered from planned track work. For the irregular weekend days, less clear reasons could be found why these days were irregular. The outcomes of this study can be applied in multiple ways. WMATA can use the methods from this study to detect disruptions per day. At the end of the day, or in the morning, WMATA can see if the day was regular or irregular. When the day was irregular, disruptions can be detected and locations that were affected can be found. Not only WMATA could use these methods, any public transport operator that has access to tap-in and tap-out data and vehicle movements can calculate the average passenger delays and use these as an input to apply the methods to detect disruptions. For a bus network this might be more complex, as buses have in general a larger headway and lower passenger counts. On top of that, buses are more severely affected by external factors, such as traffic jams. Furthermore, this study can be used to find locations where disruptions occur most often, and the total passenger delay associated with these disruptions. This can help public transport network operators to prioritise which locations to focus on to reduce passenger delays. The method in this study is not only interesting for public transport network operators, but could also be used for road networks. Speed data could be used to investigate the day-to-day regularity of road networks, to find

disruptions that are caused by accidents instead of the regular traffic jams during peak periods. Furthermore, this study could be used as a step towards real-time disruption detection, for both public transport and road networks. When the day-to-day regularity (or the regularity of data from a smaller time period) has been investigated, real-time data that deviates from this regularity can indicate disruptions in the network. This real-time data could for example be smart card data from a public transport network or speed data from a road network.

References

Briand, A., Toqué, F., Côme, E., and Oukhellou, L. (2018). Automatic detection of atypical events in a public transport network using smart card data. *Unpublished preprint*.

Ji, T., Fu, K., Self, N., Lu, C.-T., and Ramakrishnan, N. (2018). Multi-task learning for transit service disruption detection. In *2018 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining (ASONAM)*, pages 634–641. IEEE.

Krishnakumari, P., Cats, O., and Van Lint, H. (2019). Day-to-day and seasonal regularity of network passenger delay for metro networks. *Unpublished manuscript*.

Wang, Z., Bovik, A. C., Sheikh, H. R., Simoncelli, E. P., et al. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE transactions on image processing*, 13(4):600–612.

WMATA (2019a). Metro's improved 'rush hour promise' begins tomorrow. https://www.wmata.com/about/news/2019-Rush-Hour-Promise.cfm.

WMATA (2019b). Wmata metro snapshot. https://www.wmata.com/about/upload/2019-Metro-Snapshot-Fact-Sheet.pdf.

WMATA (2019c). Wmata metrorail pocket guide. https://www.wmata.com/rider-guide/new-riders/upload/pocket-guide-English.pdf.