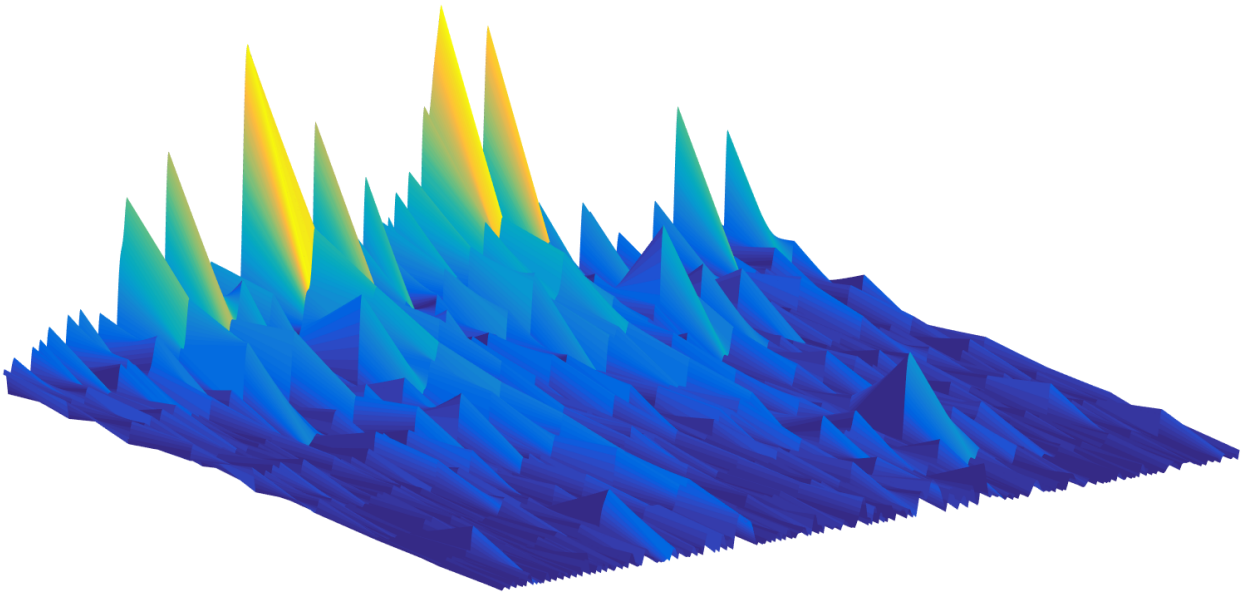
CHALMERS
UNIVERSITY OF TECHNOLOGY



Towards Improved Video Streaming from Repetitively Relocating Transmitters

Real-time available bandwidth prediction based on ARIMA models and bandwidth maps defined through Gaussian process regression

Master's Thesis in Engineering Mathematics and Computational Science

CARL SÖDERPALM

Department of Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2019

Master's Thesis in
Engineering Mathematics and Computational Science

Towards Improved Video Streaming from Repetitively Relocating Transmitters

Real-time available bandwidth prediction based on ARIMA models and bandwidth maps defined through Gaussian process regression

CARL SÖDERPALM

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Communications, Antennas, and Optical Networks
Communication Systems group
Chalmers University of Technology
Gothenburg, Sweden 2019

Towards Improved Video Streaming from Repetitively Relocating Transmitters
Real-time available bandwidth prediction based on ARIMA models and bandwidth maps defined through Gaussian process regression
CARL SÖDERPALM

Company supervisor: Dr. Per Hallgren, Einride
Academic supervisor: Dr. Chayan Bhar, Department of Electrical Engineering
Examiner: Prof. Giuseppe Durisi, Department of Electrical Engineering

Master's Thesis in Engineering Mathematics and Computational Science
Department of Electrical Engineering
Division of Communications, Antennas, and Optical Networks
Communication Systems group
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: The change of a bandwidth map converges towards zero as more and more data traces are considered during the Gaussian process regression.

Typeset in LaTeX
Printed by Chalmers Reproservice
Gothenburg, Sweden 2019

Towards Improved Video Streaming from Repetitively Relocating Transmitters
Real-time available bandwidth prediction based on ARIMA models and bandwidth maps defined through Gaussian process regression
CARL SÖDERPALM
Department of Electrical Engineering
Chalmers University of Technology

Abstract

Video quality is of utmost importance when a vehicle is driven remotely. However, long distance remote driving over wide area wireless networks may be subject to significant variations in available network bandwidth. To proactively deal with packet loss during real-time video streaming in such a scenario, the encoding bitrate of the video can be adjusted to suit upcoming network conditions. To do this, accurate prediction of future available bandwidth is necessary. In a repetitive setting, where the vehicle is driven back and forth along the same route, such predictions can utilize bandwidth maps. These maps are defined by deterministic features extracted from historic throughput measurements. In this thesis, a new method for calculating bandwidth maps, based on Gaussian process regression, is introduced. Moreover, a new time series model incorporating a bandwidth map is described and shown to outperform classical time series models in predictive accuracy when forecasting throughput. The model is shown to have a root mean squared error close to 1 Mbit/s in a network with throughput between 0 and 8 Mbit/s and fluctuations of more than 3 Mbit/s between measurements, about 18% better than a classical model. Lastly, an algorithm for bitrate planning, utilizing the above model, is developed. This results in a mean channel usage of around 65%, if the algorithm's failure rate (i.e., how often the throughput is less than the suggested bitrate) is set to be 5%. Furthermore, these results are shown to be improvable with more data.

Keywords: real-time video streaming, throughput prediction, bandwidth maps, Gaussian process regression, ARIMA.

Acknowledgements

I would like to extend my appreciation to Chayan Bhar and Per Hallgren for supporting me throughout the writing of this Master's thesis. Moreover, they and Giuseppe Durisi have my deepest gratitude for taking on this project during the summer and spending some of their vacation conversing with me over the Internet. I would also like to sincerely thank Henk Wymeersch for suggestions regarding data collection and Gaussian process regression, our discussion was very fruitful.

I am further grateful for the opportunity Einride has given me. They welcomed me into their company, took me with them on trips to test their T-Pod, and invited me to different company events during my thesis work. This firsthand experience with an exciting and hyped startup is something I will never forget.

I would like to send some love to my girlfriend, family, and friends, for always being there. You have taken care of me when I have not managed it myself and believed in me from the very beginning, which means more than you know.

Lastly, Länkspress, I will forever remember the lunches together with you all, which made the time at Chalmers worth the grind. The adventures, laughs, and horrible courses we experienced together have formed friendships that will last a lifetime. Thanks for the memories.

Carl Söderpalm, Gothenburg, August 2019

Contents

List of Figures

List of Tables

List of Acronyms

ISP Internet Service Provider
TCP Transmission Control Protocol
IP Internet Protocol
UDP User Datagram Protocol
ABW Available Bandwidth
WWAN Wireless Wide Area Network
HTTP Hypertext Transfer Protocol
PTMTP Past Tells More Than Present
ACVF Autocovariance Function
ACF Autocorrelation Function
PACF Partial Autocorrelation Function
ARIMA Autoregressive Integrated Moving Average
GPR Gaussian Process Regression
GP Gaussian Process
ANN Artificial Neural Networks
SARIMA Seasonal ARIMA
PARIMA Positional ARIMA
ARCH Autoregressive Conditional Heteroskedasticity
GARCH Generalized ARCH
RMSE Root Mean Squared Error
RSRP Reference Signals Received Power
TFRC TCP Friendly Rate Control
BBR Bottleneck Bandwidth and Round-trip propagation time
RA Reactive Algorithm
PA PARIMA Algorithm
AA ARIMA Algorithm
MA Map Algorithm
NA Norwegian Algorithm
MCU Mean Channel Usage
MLE Maximum Likelihood Estimation
AIC Aikaike's Information Criterion
AICc Aikaike's Corrected Information Criterion

1
Introduction

This chapter first introduces the industry partnering company of the thesis and describes the research problem addressed by the project. This is followed by a background section that describes the origin of the problem, the aim of the project and some limitations. The chapter concludes with a discussion about related work and novel contributions of this thesis as well as an outline of the report.

1.1 Problem description

Einride develops a transportation system based on autonomous vehicles. For safety reasons, this system includes technology for long-distance remote driving. When a vehicle is driven remotely, the quality of the video presented to the remote driver is extremely important: the more detailed the video is, the more likely the driver is to avoid accidents. However, an increase in details implies an increase in bits needed to represent the video. If the connection between the vehicle and the driver is too weak to transmit a video of a certain quality, packet loss can occur (see Section 1.2). When this happens, the quality of the video presented to the driver can be reduced to an unacceptable level. A common approach for dealing with packet loss is using loss recovery schemes [1], a reactive approach. However, Einride operates repetitive logistic flows for its customers. The problem of this thesis is developing an algorithm that utilizes this repetitiveness to predict upcoming connection speeds and suggest suitable video qualities accordingly. This is in order to proactively avoid packet loss, while still streaming with as high quality as possible.

1.2 Background

The Internet is a huge computer network, which connects an ever-increasing range of different devices called *hosts* or *end systems*. It does this through *communication links* and *network devices*, e.g., *packet switches*. These exist in different shapes and forms and determine *transmission rates* and how *packets* are forwarded throughout the network, respectively. In this thesis, transmission rates are measured in bits per second. Packets are created by end systems through segmenting data before transmission and act as the fundamental unit of information sent over the Internet. Figure 1.1 shows a graphical representation of the Internet, in which the acronym for *Internet Service Provider (ISP)* is used.

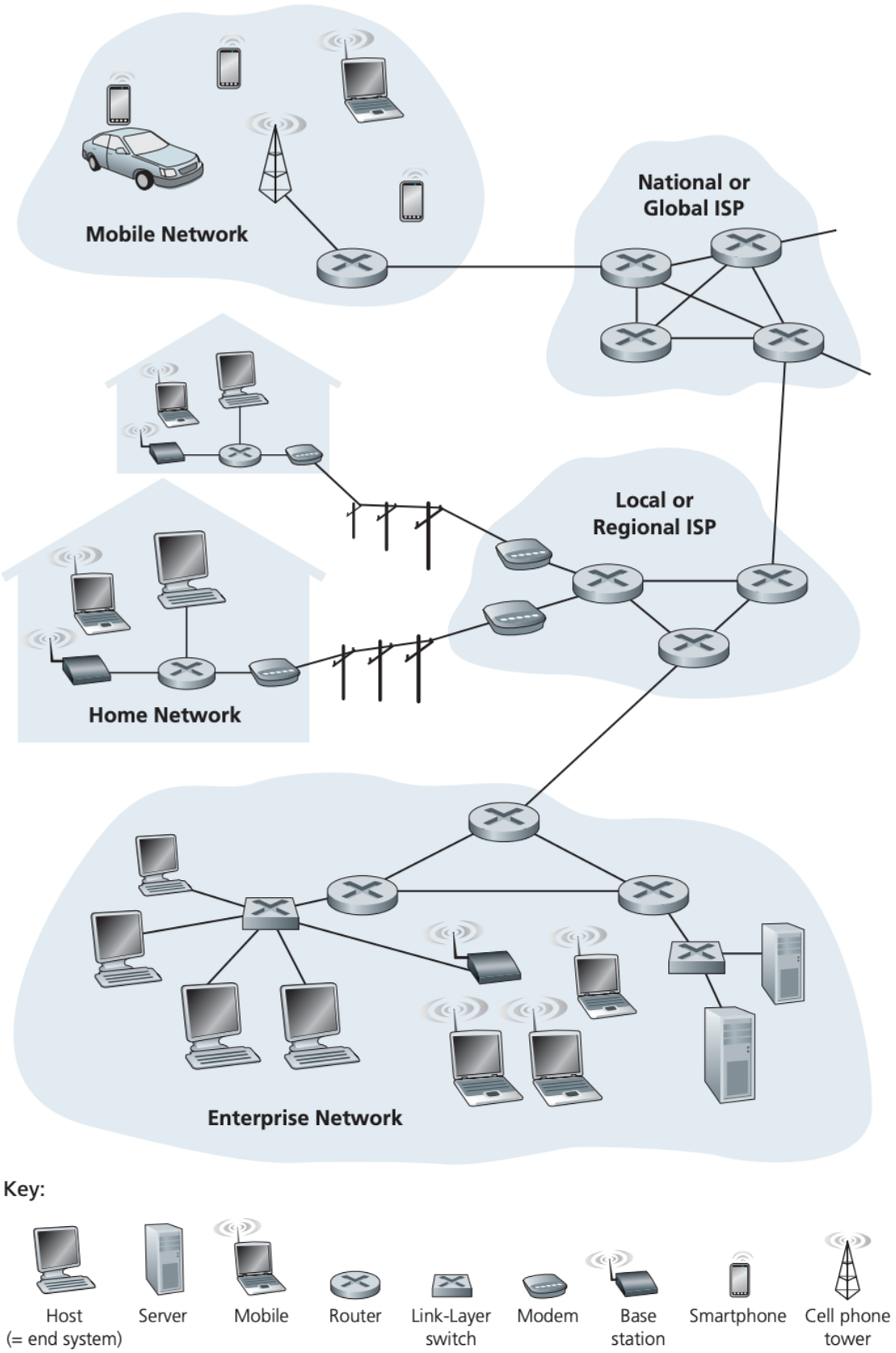


Figure 1.1: Some pieces of the Internet [1].

All communication over the Internet is governed by different *protocols*. A high level definition of protocols is [1]:

> *A protocol defines the format and the order of messages exchanged between two or more communicating entities, as well as the actions taken on the transmission and/or receipt of a message or other event.*

The *Transmission Control Protocol (TCP)* and the *Internet Protocol (IP)* are two of the most famous protocols used by the Internet. The latter specifies the fields in the network-layer packets (commonly called *datagrams*) and how end systems and network devices act on these. This protocol is the glue that binds the Internet together and is very important. For this thesis, however, the difference between TCP and the *User Datagram Protocol (UDP)*, the two transport protocols, is of great importance.

A packet switch within a packet switched network can be connected to many other switches, hosts, and end systems through multiple communication links. Moreover, every switch has a range of *output queues*, one for each link, which stores packets to be sent over that link. *Packet loss* occurs when a packet arrives at a full queue. Which packet is lost depends on the implementation of the queue, but a loss of information is certain. Therefore, packet loss is in a direct relation with the level of congestion in the network [1]. Figure 1.2 depicts the inflow of packets to the queue of a switch and how packet loss can happen when the rate of inflow is higher than the possible outflow rate.

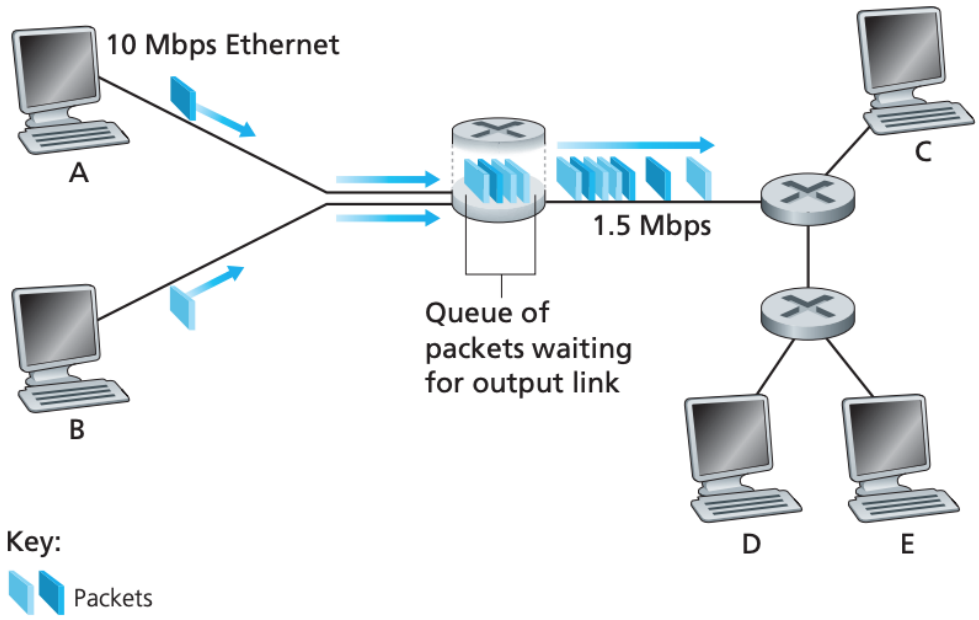


Figure 1.2: Packet switching of a switch for which the potential inflow is greater than the possible outflow [1].

Applications connected through TCP enjoy built-in congestion control, among other mechanisms that this protocol automatically employs. UDP, on the other hand, is a lightweight and connectionless oriented protocol that has no such underlying congestion-control mechanisms. Other than congestion control, TCP also has handshaking, retransmission of lost packets, flow control, etc. [2]. Excluding these type of features is clearly a double-edged sword: UDP is the protocol of choice for many real-time applications, like live video streaming, due to the lack of retransmission delays, but applications connected through it are especially vulnerable to packet loss. Since Einride has chosen to use UDP in their real-time video streaming, they have to deal with packet loss themselves.

Besides packet loss, there are two other critical performance measures for computer networks: *delay* and *throughput*. While delay certainly is a crucial measurement for real-time applications, the methods in this thesis does not work with this measurement and a longer discussion about it is therefore left out. It is however worth mentioning that there exist many different types of possible delays within a network, shown in Figure 1.3. During a transmission between two end systems, all of these contribute to the total *end-to-end delay* [1], often called *latency* or *ping* (although, sometimes these terms refer to the *round-trip delay*). Throughput, however, is central to the thesis: it is the measurement that corresponds to the speed of a connection, mentioned in Section 1.1. All of these measures are related and are simply different ways to evaluate the state of the connection between two end systems. To predict one, like throughput, and adjust a system accordingly is therefore a feasible approach when the objective is to minimize another, like packet loss.

Throughput is measured in bits per second and represents the rate of successfully transferred bits of information. The term is often used interchangeably with *bandwidth*, but the latter is actually a misnomer, as this is essentially synonymous to *maximum throughput*, for which there also exist different types [3]. The measure

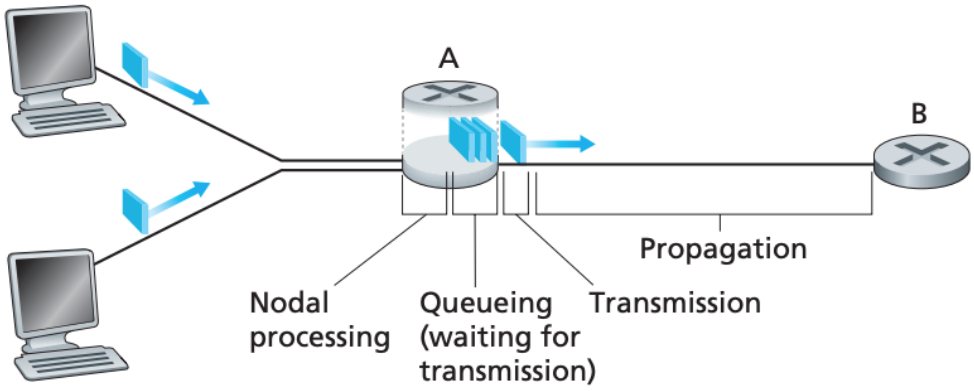


Figure 1.3: Delay occurs for different reasons; nodal processing, queueing, transmission, and propagation [1].

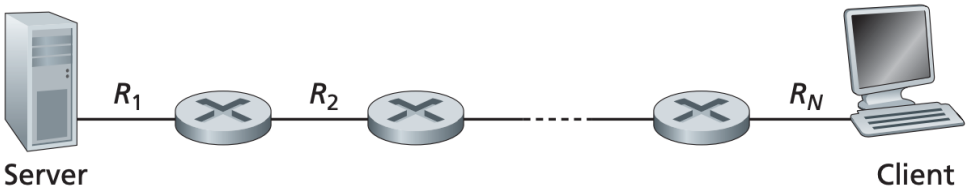


Figure 1.4: A connection with multiple links $\{R_i\}_{i=1}^{N}$ [1].

this thesis is interested in predicting is instead the *Available Bandwidth (ABW)*, i.e., the expected throughput, in the near future. This varies at all times and depends on the rate of traffic in the network. Moreover, a connection between two end systems often travels through multiple links, depicted in Figure 1.4, all possibly subjected to different levels of intervening traffic. Therefore, there will always be one that is the *bottleneck link*, defining the overall throughput of the connection. In a *Wireless Wide Area Network (WWAN)*, the ABW experienced by a moving end system varies even more, due to the change of *base stations*, environmental factors, etc. This results in both stochastic and deterministic features of ABW, illustrated and discussed in more detail in Chapter 3.

Bits per second is also a crucial measurement for video encoding, the process of reducing the amount of data required to represent a digital video signal. The amount of data used to represent the video is determined by the *bitrate* of the encoding, which refers to the number of bits used to represent the video over a certain time span (commonly one second) [4]. For real-time video streaming, the bitrate is the *goodput* required to avoid interruptions. Goodput is a portmanteau of good and throughput and is the application-level throughput of a communication, i.e., the number of useful information bits delivered by the network to a certain destination per unit of time. The amount of data considered excludes protocol overhead bits as well as retransmitted data packets. Hence, for UDP, goodput is almost equal to throughput [1]. Because of this, a suitable encoding bitrate of a real-time video streamed over UDP can be determined using ABW.

1.3 Aim

The aim of this thesis is to solve the problem stated in Section 1.1, i.e., present an algorithm that predicts ABW and uses this knowledge to specify a suitable bitrate for the real-time video streaming. In light of Section 1.2, this section expresses the purpose of such an algorithm by defining a high level optimization problem.

The stochastic nature of ABW makes it impossible to predict an upcoming level of ABW with absolute accuracy. An alternative is to design a heuristic algorithm with a low prediction error. To let this algorithm predict the upcoming ABW is, however, a mistake: if the predicted ABW is greater than the actual ABW, con-

gestion and packet loss will occur. With the goal to minimize packet loss, a more sensible objective is to predict a lower bound that the ABW is expected to be greater than with a high probability. This lower bound can then serve as the bitrate for the video encoding. The optimization problem of consideration is thus

$$\text{maximize} \quad \sum_{t \in N} B(t) \tag{1.1}$$

$$\text{subject to} \quad \min_{t \in N} \Big\{ B(t) \Big\} \geq b \tag{1.2}$$

$$\varepsilon / n \leq r, \tag{1.3}$$

where $N := \{1, 2, ..., n\}$, for n discrete time points, and $B(t)$ is the lower bound at time $t \in N$, with b as the minimum allowed value of $B(t)$, $\forall t$. Furthermore, ε is the expected amount of time points where the lower bound is not low enough, i.e., $\varepsilon := \mathbb{E}|\{t \in N : T_t < B(t)\}|$, where T_t is a stochastic variable that represents the throughput at time $t \in N$. Lastly, r is the acceptable rate of failure.

To solve this optimization problem, ε has to be estimated. If the performance of an algorithm for finding $B(t)$ is assumed to be constant over time, then the average failure rate is a constant that can be estimated by the mean performance on previous data. Then it is clear that this problem does not have a solution when $\mathbb{E}|\{t \in N : T_t < b\}| > \varepsilon$. Therefore, such cases will not be considered, meaning it is assumed that a route has been chosen such that it is feasible for the application.

1.4 Limitations

Issues regarding improved video compression, potential improvements in wireless computer communication technology, optimization of communication mast placements, predictive image enhancement from previous images, and optimization in the camera and display layers are not considered. Furthermore, the utilization of adaptive forward error correction as loss recovery is not investigated.

The analysis within the thesis is also subject to major limitations. Firstly, the algorithms cannot be tested in field due to system limitations. Hence, no tests on the actual improvement in video quality are presented. There are, however, multiple articles in the literature, for example [5, 6], which show the usefulness of techniques like the ones in this thesis. Secondly, the amount of data is limited (see Chapter 3), but enough for the results presented in Chapter 5.

1.5 Related work

The contents of this thesis belongs to the field of anticipatory networking, where different patterns observed in networks are utilized to enhance their performances. The techniques used in this field include many different approaches, such as user mobility prediction, but also forecasting the state of a network in terms of congestion, etc. With so many widely different approaches, it is useful to divide them

into subcategories, something that is done in [7]. This survey, from 2017, gives a broad overview of the field and the interested reader is referred to it for the full picture. The rest of this section will therefore focus on similar approaches to the one presented in this thesis.

Bandwidth maps attained from geotagged data of historic throughput levels were first proposed in [8] and in a subsequent article [9], which expanded on the former. In these articles, the idea of using a GPS-based bandwidth-look-up service to predict upcoming ABW was introduced and shown to be effective for improving video streaming quality. However, this was done in the setting of buffered streaming through the *Hypertext Transfer Protocol (HTTP)* over TCP, using a 3G-network, instead of the UDP- and 4G-setting of this thesis.

The previously mentioned article spurred more research into the area and several more articles, such as [10, 11, 12, 13], worked with similar datasets of geotagged ABW. In [10], the construction and usage of bandwidth maps was studied intensively. The usefulness of these type of look-up services for video streaming was yet again stressed and the *Past Tells More Than Present (PTMTP)* property was introduced. This compared the predictive performances of geotagged past ABWs and the current ABW for predicting an upcoming ABW. The methodologies behind the construction of bandwidth maps in [9] and [10] will be discussed in Chapter 3.

Predicting throughput in a more general setting, without utilizing bandwidth maps, is a well studied problem. Numerous algorithms of various levels of sophistication have been proposed for this purpose [14, 15, 16, 17, 18]. This thesis will focus on techniques from classical time series analysis, which has been successfully used in multiple articles to predict upcoming ABW [19, 20, 21, 22, 23]. Many of these, such as [19], focus on the usefulness of such methods, whereas others compare them to more sophisticated machine learning methods. In [24], a lower empirical prediction bound for throughput is established using the theory of differential entropy in information theory. Multiple algorithms are then compared to this bound and each other. Among these algorithms are ARIMA-based ones, but also methods based on *Artificial Neural Networks (ANN)*, etc. The former are shown to be very effective relative to their computational complexity.

Accurate real-time estimation of ABW is not a trivial task and has been studied extensively [25, 26, 27]. This is not a research interest of this thesis, as Einride already has a system for this, but the problem that it implies is worth knowing about. Since all logged levels of ABW will be due to a system like this, any predictions made through this data will depend on the accuracy of the system.

1.6 Novel contributions

The novelty within this thesis is threefold. It includes a novel approach, to the author's knowledge, for defining bandwidth maps based on Gaussian process regression (Section 2.6 and Section 3.3). Moreover, a new time series model tailored

for the problem of this thesis is introduced (Section 3.4). It essentially combines the bandwidth map-based and ARIMA-based methods mentioned in Section 1.5 to model ABW. Lastly, a lightweight real-time algorithm, which utilizes the previously mentioned model, is presented. It suggests a suitable encoding bitrate by estimating the lower bound $B(t + h)$ for a given lag h time steps away from current time t (Section 4.3). These three novelties in turn give rise to novel results and discussions regarding the convergence of bandwidth maps, the usefulness of position-based ABW-predicting algorithms, and some insights regarding how to choose data when constructing a bandwidth map.

1.7 Outline

The rest of the thesis is divided into five chapters. First, some necessary theory regarding time series analysis will be tackled. Second, the method for data collection will be introduced together with an analysis of the data, a discussion regarding bandwidth maps, and the introduction of a novel time series model. Third, the algorithm that was developed during the project will be presented together with problems faced by it and alternative algorithms. Fourth, the various results of the thesis will be discussed, with a focus on the performance of the different stages of the proposed algorithm. Fifth, some conclusions consisting of remarks and suggestions for future work will be given.

2
Time Series Analysis

The following chapter is focused on the analysis of time series. It begins with some necessary preliminaries and introduces the reader to the classical way of analyzing this type of data through ARIMA models. This is followed by three sections describing the details of this method and statistical tests that will be used throughout the thesis to evaluate models and data characteristics. After this, a more general way of modeling time series, that of state space models, is briefly discussed. Lastly, Gaussian process regression is introduced as an alternative to Kalman smoothing for finding an underlying state within noisy data.

Sections 2.1 and 2.2 are based on the standard introductory book [28], *Introduction to Time Series and Forecasting*, by Peter J. Brockwell and Richard A. Davis, which the interested reader is referred to for more details. Moreover, definitions given throughout these sections are the same as the corresponding definitions found in [28], for the sake of consistency.

2.1 Preliminaries

A *time series* is a set of observations $x_1, ..., x_n$ ordered in time. These are usually equally spaced in a discretized time domain and exhibit some type of stochastic behaviour. In the words of Brockwell and Davis [28]:

> *A time series model is a specification of the joint distributions of a sequence of random variables, of which a sequence of observations is postulated to be a realization.*

This means that a time series model consists of the formulation of a stochastic process in an attempt to describe how the time series came to be. However, the distinction between the data and the underlying process is often dropped in the literature and the phrase *time series* is used interchangeably.

White noise,

$$\{X_t\} \sim \mathrm{WN}(0, \sigma^2), \quad t \in \mathbb{N}, \tag{2.1}$$

for example, is a sequence of uncorrelated random variables, each with zero mean and variance σ^2. This is a discrete stochastic process with an infinite set of possible time series realizations. One such realization can be seen in Figure 2.1. It is

a suitable choice for a first example, as it possesses an important feature for the analysis of time series: *stationarity*. To define stationarity, the concepts of mean and covariance functions need to be introduced.

Definition. If $\{X_t\}$ is a time series with $\mathbb{E}(X_t^2) < \infty$, then the **mean function** of $\{X_t\}$ is

$$\mu_X(t) = \mathbb{E}(X_t). \tag{2.2}$$

□

Definition. For the same time series as above, the **covariance function** is

$$\gamma_X(r, s) = \mathrm{Cov}(X_r, X_s) = \mathbb{E}[(X_r - \mu_X(r))(X_s - \mu_X(s))], \tag{2.3}$$

for all integers r and s. □

Definition. The time series $\{X_t\}$ is (**weakly**) **stationary** if $\mu_X(t)$ is independent of t and $\gamma_X(t + h, t)$ is independent of t for each h. □

With these definitions, it is easy to verify the statement about the stationarity of white noise. A *random walk* defined by the cumulative sum of a white noise, seen in Figure 2.1, is an example of a nonstationary time series. To analyze such a time series in a classical manner, achieving stationarity through some sort of transformation is necessary. In fact, this is one of the first steps of the so called Box-Jenkins method, an iterative modeling approach for time series analysis, which classically consists of three steps. The first step utilizes the following definitions.

Definition. Let $\{X_t\}$ be a stationary time series. The **Autocovariance Function (ACVF)** of $\{X_t\}$ at lag h is

$$\gamma_X(h) = \mathrm{Cov}(X_{t+h}, X_t), \tag{2.4}$$

□

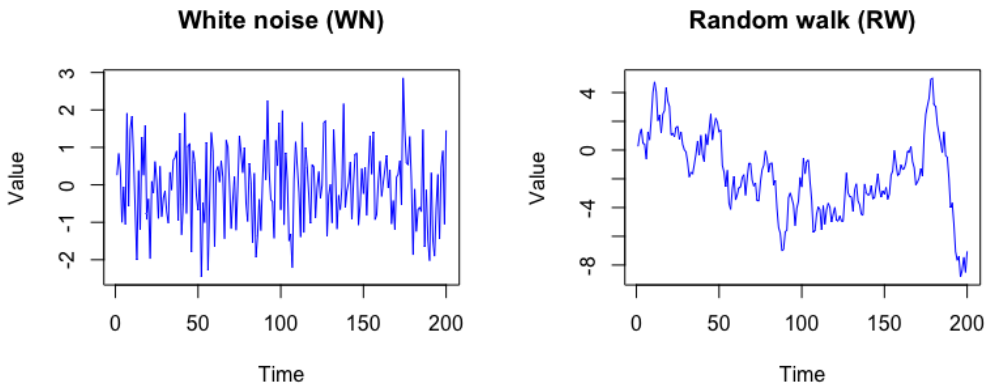


Figure 2.1: Gaussian white noise (to the left) and its cumulative sum, a random walk (to the right).

Definition. For a stationary $\{X_t\}$, the **Autocorrelation Function (ACF)** at lag h is

$$\rho_X(h) = \frac{\gamma_X(h)}{\gamma_X(0)} = \text{Cor}(X_{t+h}, X_t). \tag{2.5}$$

□

Definition. Let $\{X_t\}$ yet again be stationary, then the **Partial Autocorrelation Function (PACF)** at lag h is

$$\alpha(h) = \text{Cor}[X_t - P(X_t|X_{t+1}, ..., X_{t+h-1}), X_{t+h} - P(X_{t+h}|X_{t+1}, ..., X_{t+h-1})], \tag{2.6}$$

where $P(Y|Z)$ is the orthogonal projection of Y onto a linear subspace of the Hilbert space spanned by Z. This means that it accounts for the values in-between X_t and X_{t+h} and as such is a conditional correlation. □

Definition. Let $x_1, ..., x_n$ be observations of a time series. The **sample mean** is

$$\bar{x} = \frac{1}{n}\sum_{t=1}^{n} x_t, \tag{2.7}$$

the **sample covariance function** is

$$\hat{\gamma}(h) = n^{-1}\sum_{t=1}^{n-|h|}(x_{t+|h|} - \bar{x})(x_t - \bar{x}), \quad -n < h < n, \tag{2.8}$$

and the **sample autocorrelation function** is

$$\hat{\rho}(h) = \frac{\hat{\gamma}(h)}{\hat{\gamma}(0)}, \quad -n < h < n. \tag{2.9}$$

□

These functions are often presented in graphs called *corellograms*. In Figure 2.2, the sample ACF of the white noise and the random walk in Figure 2.1 are plotted. The lack of any autocorrelation in the white noise is clearly visible, while the opposite is true for the random walk.

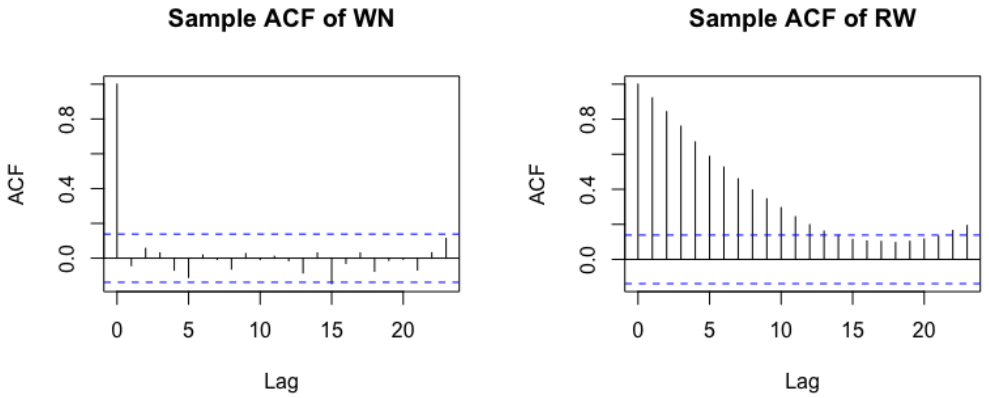


Figure 2.2: An example of ACF plots. Lags with values within the blue dashed lines are uncorrelated with a confidence level of 95%.

With the above in mind, the Box-Jenkins method is as follows:

1. Achieve stationarity through necessary transformations, then investigate the stationary data through its ACF and PACF.
2. Using optimization, find the parameters and coefficients of the model that best fit the data.
3. Determine if the model is reasonable through statistical tests.

A beginner in the field might at this point wonder what these transformations are, how the optimization is carried out, and which statistical tests are used. The following three sections tackle these questions, with each section especially related to the corresponding step in the method.

2.2 ARIMA models

The first step of the Box-Jenkins method is concerned with finding an appropriate *Autoregressive Integrated Moving Average (ARIMA)* model for the data. Such a model has three hyperparameters, which determine the structure and order of the model, and is usually denoted ARIMA(p, d, q).

If $d \neq 0$, there is a d-differencing in the model. A 1-differencing of a time series $\{X_t\}$ results in a new time series $\{Y_t\}$ defined by

$$Y_t = X_t - X_{t-1}, \quad \forall t \in \mathbb{Z}. \tag{2.10}$$

A 2-differencing of $\{X_t\}$ is the 1-differencing of $\{Y_t\}$. This is often expressed in terms of the *difference operator*, Δ, or the *lag operator*, L, defined as

$$\Delta X_t = (1 - L)X_t = X_t - X_{t-1}. \tag{2.11}$$

With these operations, a d-differencing of X_t is easily written as $\Delta^d X_t$ or $(1-L)^d X_t$. An example of this transformation is a 1-differencing of the random walk defined earlier, which would result in the underlying white noise (Figure 2.1).

Simple d-differencing is useful when trying to remove linear trend and is one of three common ways to transform a time series to achieve stationarity. The other common transformations are applying a logarithmic function to the data, to deal with non-linear trend, and differencing due to a known seasonal length, to deal with evenly spaced recurring patterns. (Monthly spaced data points are for example often differenced with their 12^{th} predecessor, an operation commonly denoted Δ_{12}.) The latter of these is known as a *Seasonal ARIMA (SARIMA)* model, one of a multitude of available extensions to ARIMA. Another one of these, the *Autoregressive Conditional Heteroskedasticity (ARCH)* model for variable variance, will be mentioned in Section 2.4, together with a discussion on stationarity testing, i.e., how to decide if the transformations used are sufficient.

The proper definition of an ARIMA(p, d, q) process $\{X_t\}$ is that the process $\{Y_t\} =$

$\{\Delta^d X_t\}$ is a (causal) ARMA(p, q) process. An ARMA(p, q) process is a stationary time series consisting of two different regressions, AR(p) and MA(q). In an AR(p) process, each value at a given time point is given by regressing on the p values before it, hence the name autoregressive. This is commonly written as

$$Y_t = c + Z_t + \sum_{i=1}^{p} \phi_i Y_{t-i}, \quad Z_t \sim \mathrm{WN}(0, \sigma^2), \quad \forall t \in \mathbb{Z}, \tag{2.12}$$

where c is a constant to achieve zero mean and Z_t is uncorrelated with X_s, $\forall s < t$. An MA(q) process can be thought of as a regression on the q preceding noise terms, i.e.,

$$Y_t = \mu + Z_t + \sum_{i=1}^{q} \theta_i Z_{t-i}, \quad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2), \quad \forall t \in \mathbb{Z}, \tag{2.13}$$

where μ is the expectation of Y_t. Figure 2.3 shows an AR(1) process and an MA(1) process. In this picture, both are stationary processes, but the AR process is close to being nonstationary, which occurs when $|\phi| \geq 1$.

Putting AR(p) and MA(q) together gives the formulation for the celebrated ARMA(p, q) process,

$$Y_t = C + Z_t + \sum_{i=1}^{p} \phi_i Y_{t-i} + \sum_{i=1}^{q} \theta_i Z_{t-i}, \quad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2), \quad \forall t \in \mathbb{Z}. \tag{2.14}$$

The above is sometimes expressed more concisely as $\phi(L, p)Y_t = \theta(L, q)Z_t$, where the constant has been dropped, assuming zero mean, and with the operator polynomials $\phi(L, p) = 1 - \sum_{i=1}^{p} \phi_i L^i$ and $\theta(L, q) = 1 + \sum_{i=1}^{q} \theta_i L^i$. This then leads to the final expression for an ARIMA(p, d, q) process $\{X_t\}$:

$$\phi(L, p)\Delta^d X_t = \theta(L, q)Z_t, \quad \{Z_t\} \sim \mathrm{WN}(0, \sigma^2), \quad \forall t \in \mathbb{Z}. \tag{2.15}$$

Having the above theory established enables a discussion on the manual selection of p and q. As mentioned, this is commonly done through inspection of the sample

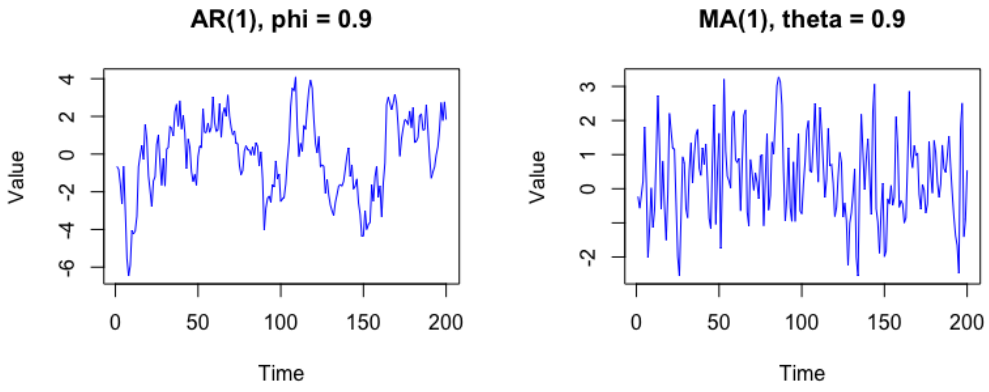


Figure 2.3: Realizations of an AR(1) process an MA(1) process.

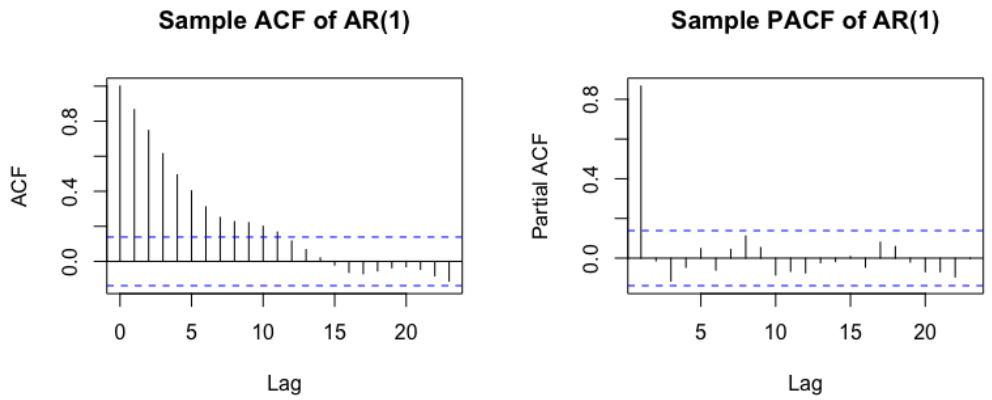


Figure 2.4: The ACF and PACF of the AR(1) realization displayed in Figure 2.3. Note that the PACF created by the R function *pacf* does not show lag 0.

ACF and PACF of the time series. Intuitively, the ACF could be seen as portraying the *complete* autocorrelation between two data points; it includes all the underlying structures that cause correlation. The PACF is *partial* and removes these structures, simply showing the direct correlation between the two points.

With this insight, the correlograms of a realization of an AR(p) process are quite easily understood. A strong correlation in the ACF up to p is to be expected, followed by a slow decline due to the inertia created by earlier correlations. The PACF, on the other hand, should rapidly decline and settle around a correlation of zero after p. This can be seen in Figure 2.4.

A realization of an MA(q) process exhibits the opposite behaviour (seen in Figure 2.5), but this is not as easily grasped. Its ACF cuts off after q, because all noise terms are uncorrelated, meaning each time point is only correlated with other time points affected by the same noise terms. The behaviour of the PACF is best explained by *invertibility*. As long as the coefficients θ_i, $i = 1, 2, ..., q$, of an MA(q) process are within the unit circle, there is invertibility, meaning that it can be *inverted* to an infinite AR process. This means that the PACF of this MA(q) process is really the

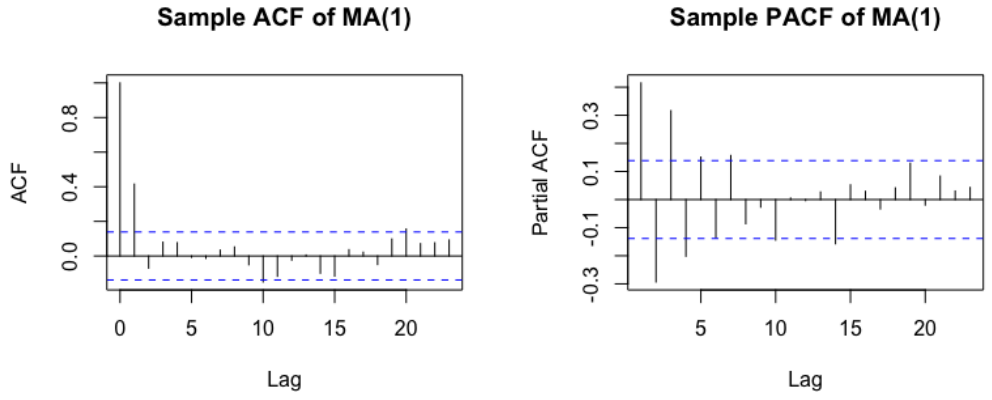


Figure 2.5: The ACF and PACF of the MA(1) realization displayed in Figure 2.3.

PACF of an AR(∞) and therefore trails off into infinity. The interested reader is referred to [28] for a deeper understanding of this concept and the proofs behind it. Invertibility is furthermore the dual concept to the already mentioned *causality*, which holds when all ϕ_i, $i = 1, 2, ..., p$, are within the unit circle.

How to use the above knowledge for the manual selection of p and q is summarized in Table 2.1. Clearly, the inspection of ACF and PACF for estimating p and q is not always useful. In fact, Brockwell and Davis suggest the use of information criteria (Section 2.3) for order specification of mixture models. However, these require maximization of likelihood functions and are therefore not the most efficient routes to take if a pure model of only AR or MA seems sufficient.

	AR(p)	MA(q)	ARMA(p, q)
ACF	Decays to 0	Cuts off after q	Decays to 0
PACF	Cuts off after p	Decays to 0	Decays to 0

Table 2.1: Summary of ACF and PACF behaviours for different processes.

2.3 Parameter estimation and model selection

The second step in the Box-Jenkins method consists of estimating the parameters in (2.14). This section is kept relatively short, leaving out intricate technical details. Instead, it will serve as a high level description of the different methodological alternatives for estimating these parameters and give an understanding of what happens under the hood in available solvers.

If the model is a pure AR process, the *Yule-Walker equations*,

$$\gamma_X(h) = \sum_{i=1}^{p} \phi_i \gamma_X(h-i) + \sigma^2 \delta_{h,0}, \quad h = 0, ..., p, \tag{2.16}$$

where $\delta_h, 0$ is the *Kronecker delta function*, can be solved efficiently through various algorithms. As seen in (2.16, these are a system of linear equations consisting of the coefficients and the ACVF of the process. When put into matrix form, the equations with $h > 0$ involve a *Toeplitz matrix*, which allows for the use of the *Durbin-Levinson algorithm* instead of the less efficient *Gauss-Jordan elimination*. After this, the last equation of $h = 0$ can easily be solved for σ^2. If the model is purely MA, there is the similar *innovations algorithm* [28].

Using the Yule-Walker equations for mixture models, known as the *method of moments*, is cumbersome and leads to inefficient estimates [29]. Instead, several other approaches are used, all with their own strengths and weaknesses. A common first step is to estimate starting values for the optimization procedure through for example the *Hannan-Rissanen algorithm*, which involves the use of a high order AR model and the Yule-Walker estimates for it [30].

When first estimates have been attained, a common approach is using an ordinary non-linear least squares optimization, for which the objective is minimizing the sum of squared residuals (Section 2.4). This is useful, since it requires no assumptions about the distribution of the *innovations*, i.e., one-step prediction errors. However, when these can be assumed to be Gaussian, *Maximum Likelihood Estimation (MLE)* is the weapon of choice, as it usually provides the most accurate estimates [28]. Such an assumption can often be made, and MLE is therefore the default fitting mechanism of for example the *forecast-package* in R [31, 32]. If $\{X_t\}$ is a Gaussian time series with zero mean and ACVF $\kappa(i,j) = \mathbb{E}(X_i X_j)$, with $\mathbf{X}_n = (X_1, ..., X_n)^\top$ and $\hat{\mathbf{X}}_n$ as its conditional expectation. Then the effective calculation of the likelihood

$$L(\Gamma_n) = (2\pi)^{-n/2}(\det \Gamma_n)^{-1/2} \exp\left\{-\frac{1}{2}\mathbf{X}_n^\top \Gamma_n^{-1}\mathbf{X}_n\right\} \tag{2.17}$$

of $\mathbf{X}_n$ relies on the fact that the calculation of the determinant and inverse of the nonsingular covariance matrix Γ_n can be avoided. The proof of this can be found in [28] and results in the following expression for the Gaussian likelihood of an ARMA process:

$$L(\boldsymbol{\phi}, \boldsymbol{\theta}, \sigma^2) = \frac{1}{\sqrt{(2\pi\sigma^2)^n r_0 \dots r_{n-1}}} \exp\left\{-\frac{1}{2\sigma^2}\sum_{j=1}^{n}\frac{(X_j - \hat{X}_j)^2}{r_{j-1}}\right\}, \tag{2.18}$$

where r_i, $i = 0, ..., n$, are due to the innovation variances v_i through $r_i = v_i/\sigma^2$. The maximum likelihood estimators $\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}$, and $\hat{\sigma}^2$ are then given by the maximization of L w.r.t. $\boldsymbol{\phi}$ and $\boldsymbol{\theta}$.

The usefulness of estimating the parameters through MLE is especially great if the order selection is made through the evaluation of some information criterion. The classical one is *Aikaike's Information Criterion (AIC)*, which is biased for small sample sizes. For this reason, the corrected version, AICc is more common [28].

Definition. To apply **Aikaike's Corrected Information Criterion (AICc)**, choose $p, q, \boldsymbol{\phi}_p$, and $\boldsymbol{\theta}_q$ to minimize

$$\text{AICc} = -2\ln L\left(\boldsymbol{\phi}_p, \boldsymbol{\theta}_q, \frac{1}{n}\sum_{j=1}^{n}\frac{(X_j - \hat{X}_j)^2}{r_{j-1}}\right) + 2n\frac{p+q+1}{n-p-q-2}. \tag{2.19}$$

□

With this tool, automating the modelling of ARMA processes is simple: conduct a grid search over pre-specified spans for p and q and fit a model using each combination, then choose the most accurate model according to AICc.

2.4 Diagnostic checking

The third step of the Box-Jenkins method is concerned with model checking; the act of statistically assessing the appropriateness of a fitted model. This is done through

comparing the observed values with the corresponding predicted values. The *residuals* can be defined in different ways, but defining them in the same way as the *forecast-package* in R is the simplest, most intuitive, and easiest to work with, due to this resource.

Definition. If $\hat{\boldsymbol{\phi}}$, $\hat{\boldsymbol{\theta}}$, and $\hat{\sigma}^2$ are the estimators given by some fitting procedure applied to an ARMA(p, q) model, with the data $\mathbf{X}_n$. Then $\hat{X}_t(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}})$, $t = 1, ..., n$ is the predicted value of X_t given $X_1, ..., X_{t-1}$. The **residual** at time t is then defined as

$$R_t = X_t - \hat{X}_t(\hat{\boldsymbol{\phi}}, \hat{\boldsymbol{\theta}}) \tag{2.20}$$

and the collection of these is referred to as the **residuals** of the model. □

For a model to be valid, the properties of its residuals should reflect those of the white noise sequence generating the underlying ARMA process [28]. Therefore, the residuals should exhibit no autocorrelation and, for the assumption necessary for MLE to hold, be seemingly Gaussian. Noisy residuals also mean that all relevant information is already extracted from the data and, due to the fitting procedure, at an optimal rate. There are several ways to check if this is the case:

- Simply inspecting the time series of the residuals. If it is clearly is nonstationary or exhibits other characteristics that are not likely to be found in a noise sequence, the model is probably not sufficient [33].

- Testing for stationarity using the *KPSS test*, which tests the null hypothesis that a given time series is stationary around a deterministic trend [34].

- Analyzing the sample ACF of the residuals. Its behaviour should resemble that of a white noise process. This can be extended to fitting an AR model to the residuals through AICc. A selected order of zero would suggest white noise [28].

- Inspecting a *lag plot* can further strengthen the belief of no autocorrelation at lag $h = 1$ [33].

- Conducting a *portmanteau test*. There exist several different ones of these, but the most common ones are the *Ljung-Box test*, an improved version of the *Box-Pierce test*, and the similar *McLeod-Li test*. These test the null hypothesis that a given time series, in this case the residuals, are serially uncorrelated [28].

- Checking for normality. This can be done through inspection of the histogram of the residuals and Q-Q or empirical CDF plots. There are also normality tests, such as the *Kolmogorov-Smirnov test*, the *Shapiro-Wilk test*, and the *Jarque-Bera test*. However, tests like these are not utilized in this thesis.

The methods presented above are usually combined when assessing the appropriateness of the model. During this assessment, the discovery of *heteroscedasticity* within the residuals may occur. This is the presence of varying variance and can be further assessed by for example the *Breusch-Pagan test* [35]. To deal with this, a *Generalized ARCH (GARCH)* model needs to be used. Without the generalization, this is an AR model that incorporates noise terms that themselves resemble AR processes. The AR(p)/ARCH(r) model is defined as

$$Y_t = c + Z_t + \sum_{i=1}^{p} \phi_i Y_{t-i} \tag{2.21}$$

$$Z_t = e_t \left(\alpha_0 + \sum_{i=1}^{r} \alpha_i Z_{t-i}^2 \right)^{1/2}, \quad \{e_t\} \sim \text{IID } N(0,1), \tag{2.22}$$

with $\alpha_0 > 0$ and $\alpha_j \geq 0$, $j = 1, ..., r$. The generalization includes the necessary adjustments for ARMA processes [28]. These types of processes will not be studied in this thesis, but are worth knowing about. Note that for a process where the opposite (*homoscedasticity*) holds, there is an equivalence between the innovations of the process and the residuals of the MLE model.

2.5 State space models

With the theory of the Box-Jenkins method and ARIMA models established, it is worth noting a common, modern way to represent these types of models and work with them. *State space representations* have their origins in control engineering, where they are used to model a physical system as a set of input, output and state variables. For this brief section, the notation of [36] will be utilized. The interested reader is referred to this book for a much deeper treatment of these methods.

The general linear Gaussian state space model can be written

$$y_t = Z_t \alpha_t + \epsilon_t, \qquad \epsilon_t \sim N(0, H_t) \tag{2.23}$$

$$\alpha_{t+1} = T_t \alpha_t + R_t \eta_t, \qquad \eta_t \sim N(0, Q_t), \tag{2.24}$$

for $t = 1, ..., n$. In the *observation equation* (2.23), y_t represents a vector of *observations* at time t, and in the *state equation* (2.24), α_t represents the unknown underlying vector of *state variables*. Essentially, this models a noisy system, due to η_t, which naturally develops over time through a postulated *propagation matrix*, T_t. This system is being measured through an *observation matrix*, Z_t, an act that also introduces noise, ϵ_t.

Interestingly, the set of all possible ARIMA models is a subset of the set of all possible state space models. Hence, every ARIMA model can be put into a state space representation. This is very useful, as various filters, like the famous *Kalman filter*, can be applied to state space models. Filters like these are efficient recursive algorithms for filtering out the noise in a measured system and predicting upcoming values. This filtering and prediction is equivalent to the predictions of ARIMA models. Because of this, it has the same *reactive look* to it, like it is lagging behind the

process it is predicting (an example of this is depicted in Figure 3.10). The advantage of state space models and filters is that they provide retrospective smoothers. These correct for this lag and attain an estimate of the underlying state that caused the measurements. This is a suitable tool when the underlying state of past data is of interest for future predictions. However, for the model described above, this type of smoothing returns a *piecewise linear* estimate of the underlying state, which is not necessarily the best approximation. Therefore, this thesis will not be using the above methods other than through functions available in R that incorporate them (e.g., the calculation of the likelihood (2.18) in R's *arima* function, which is done through the Kalman filter). Instead, estimation of the underlying state will be achieved by *Gaussian process regression*.

2.6 Gaussian process regression

Gaussian Process Regression (GPR) is a way to perform *Bayesian inference* over functions using a distribution of functions. This is in opposition to the common approach in *supervised learning* of inferring over a parametric representation of some function f. This section is mainly due to Rasmussen and Williams [37], and the condensed version of this given by Murphy [38].

A *Gaussian Process (GP)* is a stochastic process that has a multivariate normal distribution for any arbitrarily chosen set of time points. The distribution of a GP is the joint distribution of all of these random variables and is therefore a distribution over functions. In the setting of Bayesian inference, this defines a prior over functions, which is converted into a posterior when given some observed data. The attentive reader might realize that this is somewhat similar to the ARIMA processes presented in earlier sections. Indeed, these processes are in the subset of GPs known as *discrete-time Gaussian Markov processes*.

If the prior of the regression function is a GP, i.e.

$$f(\mathbf{x}) \sim GP(\mu(\mathbf{x}), \kappa(\mathbf{x}, \mathbf{x}')), \tag{2.25}$$

where μ and κ are the mean and covariance functions. Then, for a finite set of points $\mathbf{X}$, this process defines the joint Gaussian $p(\mathbf{f}|\mathbf{X}) = N(\mathbf{f}|\boldsymbol{\mu}, \mathbf{K})$, for which $\boldsymbol{\mu}$ and $\mathbf{K}$ are the mean vector and the matrix due to the covariance function of $\mathbf{X}$. Moreover, $\mathbf{f}$ denotes the vector of outputs of f given $\mathbf{X}$.

Assuming a noise-free training set $\{(\mathbf{x}_i, f(\mathbf{x}_i)),\ i = 1, ..., n)$, for predicting function outputs $\mathbf{f}_*$ from a test set $\mathbf{X}_*$, means the estimated underlying function needs to interpolate the training points. Then the joint distribution is

$$\begin{pmatrix} \mathbf{f} \\ \mathbf{f}_* \end{pmatrix} \sim N\left(\begin{pmatrix} \boldsymbol{\mu} \\ \boldsymbol{\mu}_* \end{pmatrix}, \begin{pmatrix} \mathbf{K} & \mathbf{K}_* \\ \mathbf{K}_*^T & \mathbf{K}_{**} \end{pmatrix} \right), \tag{2.26}$$

where $\mathbf{K}_* = \kappa(\mathbf{X}, \mathbf{X}_*)$, etc. The posterior then has the form

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{f}) = N(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{2.27}$$
$$\boldsymbol{\mu}_* = \boldsymbol{\mu}(\mathbf{X}_*) + \mathbf{K}_*^T\mathbf{K}^{-1}(\mathbf{f} - \boldsymbol{\mu}(\mathbf{X})) \tag{2.28}$$
$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{K}_*. \tag{2.29}$$

When noisy data is considered, the observations have to be assumed to be the result of the underlying function plus a noise term, i.e., $y = f(\mathbf{x}) + \epsilon$, $\epsilon \sim N(0, \sigma_y^2)$. This means that the model is not required to interpolate the training data, but should be close enough to doing so. There is then a covariance of the observations, $\mathrm{Cov}(y_i, y_j) = \kappa(\mathbf{x}_i, \mathbf{x}_j) + \sigma_y^2\delta_{ij}$, with δ_{ij} being the Kronecker delta, meaning $\mathrm{Cov}(\mathbf{y}|\mathbf{X}) = \mathbf{K} + \sigma_y^2\mathbf{I}_n =: \mathbf{K}_y$. This alters the joint distribution, which turns out to be

$$\begin{pmatrix}\mathbf{y}\\ \mathbf{f}_*\end{pmatrix} \sim N\left(\mathbf{0}, \begin{pmatrix}\mathbf{K}_y & \mathbf{K}_*\\ \mathbf{K}_*^T & \mathbf{K}_{**}\end{pmatrix}\right), \tag{2.30}$$

and leads to the key predictive equations for Gaussian process regression

$$p(\mathbf{f}_*|\mathbf{X}_*, \mathbf{X}, \mathbf{y}) = N(\mathbf{f}_*|\boldsymbol{\mu}_*, \boldsymbol{\Sigma}_*) \tag{2.31}$$
$$\boldsymbol{\mu}_* = \mathbf{K}_*^T\mathbf{K}^{-1}\mathbf{y} \tag{2.32}$$
$$\boldsymbol{\Sigma}_* = \mathbf{K}_{**} - \mathbf{K}_*^T\mathbf{K}_y^{-1}\mathbf{K}_*, \tag{2.33}$$

which for a single test point $\mathbf{x}_*$ reduces to

$$p(f_*|\mathbf{x}_*, \mathbf{X}, \mathbf{y}) = N(f_*|\mathbb{E}(f_*), \mathrm{Var}(f_*)) \tag{2.34}$$
$$\mathbb{E}(f_*) = \mathbf{k}_*^\top\mathbf{K}_y^{-1}\mathbf{y} \tag{2.35}$$
$$\mathrm{Var}(f_*) = \kappa(\mathbf{x}_*, \mathbf{x}_*) - \mathbf{k}_*^\top\mathbf{K}^{-1}\mathbf{k}_*. \tag{2.36}$$

The above clearly indicates that the covariance function κ, or the *kernel*, as it is often called, plays a critical role. The choice of it is therefore crucial to the predictive performance of a GPR. There are several to choose from, but a common one is the *squared-exponential* kernel, also known as the Gaussian kernel,

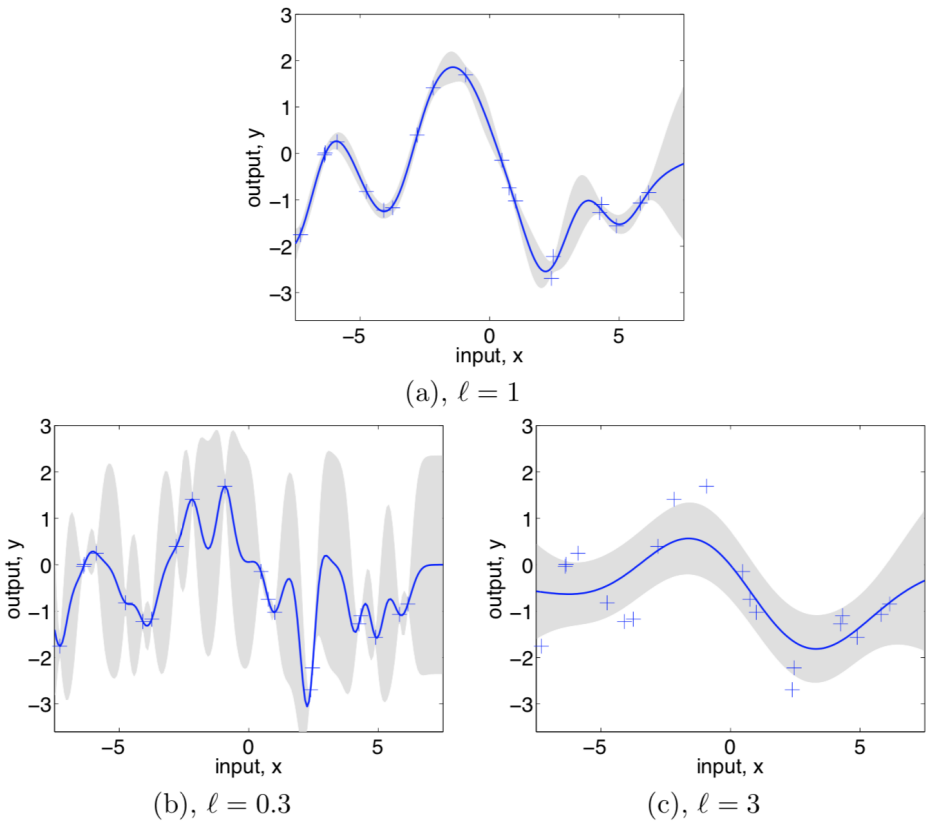
$$\kappa_y(x_i, x_j) = \sigma_f^2 \exp(-\frac{1}{2l^2}(x_i - x_j)^2) + \sigma_y^2\delta_{ij}. \tag{2.37}$$

The hyperparameters of the kernel at use specifies the shapes of the functions that will be considered. This can be seen in Figure 2.6.

The optimization procedure of GPR is the maximization of the marginal likelihood,

$$p(\mathbf{y}|\mathbf{X}) = \int p(\mathbf{y}|\mathbf{f}, \mathbf{X})p(\mathbf{f}|\mathbf{X})\,\mathrm{d}\mathbf{f}, \tag{2.38}$$

which depends on the kernel's parameters. This is realized by observing that $p(\mathbf{y}|\mathbf{X}) = N(\mathbf{f}|\mathbf{0}, \mathbf{K})$ and $p(\mathbf{y}|\mathbf{f}) = \prod_i N(y_i|f_i, \sigma_y^2)$, which implies $p(\mathbf{y}|\mathbf{X}) = N(\mathbf{y}|\mathbf{0}, \mathbf{K}_y)$. The optimization can then be done through calculating the partial derivatives of the log marginal likelihood w.r.t. the parameters of the kernel and using a standard



(a), $\ell = 1$

(b), $\ell = 0.3$

(c), $\ell = 3$

Figure 2.6: Given data generated by a GP with a Gaussian kernel and the hyperparameters $(l, \sigma_f, \sigma_n) = (1, 1, 0.1)$. (a) shows a GPR with these same parameters, while (b) shows a regression achieved by $(l, \sigma_f, \sigma_n) = (0.3, 1.08, 0.00005)$ and (c) by $(l, \sigma_f, \sigma_n) = (3, 1.16, 0.89)$.

gradient-based optimizer. However, this has the problem of nonconvexity, so there might exist nonglobal minima.

Further details about the implementation of the regression algorithm in terms of computational efficiency and a discussion about how GPR is a *linear smoother* is left out. The interested reader is referred to [37].

3

Data Collection and Analysis

This chapter describes how to collect data like the ones in this thesis. It then includes an initial analysis of these data and discusses the results of the analysis. This is followed by a discussion on the different ways to define and calculate bandwidth maps, which introduces a novel method for doing this using GPR and argues for its usefulness. The chapter concludes with the introduction of a new type of time series model, which is akin to a seasonal model, but uses differencing due to the spatial instead of the temporal domain.

3.1 Geotagged data logging

The data in this thesis is due to manual recording, using an Android app called G-NetTrack Pro, developed by Gyokov Solutions [39], along a certain route outside of Gothenburg, between Delsjömotet and Mölnlycke (Figure 3.1). The data consists of 10 distinct traces from 5 trips back and forth, i.e., 5 trips each way.

The route was chosen with its environment in mind. It exhibits many different types of environments, which are all likely to be encountered by an autonomous logistics vehicle. The speed limit never rises above 70 km/h and the environments imply driving next to a highway, on a common country road, in-between vegetation, next to a lake, through a town center, underneath a bridge, next to grocery stores, and through neighborhoods.

G-NetTrack Pro allows its user to log a multitude of parameters, ranging form GPS coordinates and movement speed, to latency, upload and download capacity,

Figure 3.1: An 11 minute drive from Delsjömotet to Nysätervägen, Mölnlycke.

Reference Signals Received Power (RSRP), etc. To log latency and upload capacity parameters, the user has to specify the corresponding URLs to use.

Since the measurements of interest in this thesis are throughput from a vehicle to a control center, GPS coordinates, and speed, these parameters are logged as a tuple approximately every second. To model the throughput, the upload capacity in G-NetTrack Pro is used, with `http://de.testmy.net/b/upload` as the URL.

3.2 Initial data analysis

There are different ways to visualize the data logged by the method explained in the previous section. A throughput measurement of a given trip could be plotted on the y-axis against time on the x-axis, like a classical time series. The second possibility is plotting it on the z-axis against an xy-plane representing the GPS coordinates. Henceforth, the former will be referred to as the *temporal representation* and the latter as the *positional representation* (*spatial* would have been suitable, but the choice of a word with a first letter different from s will become clear in Section 3.4).

Starting with the positional representation and plotting every trip in the same plot, the existence of some underlying state becomes clear (Figure 3.2). However, there seem to be areas that are more stable, with smaller throughput variances, than others (an unstable area is seen around $\Phi = 12.1$, $\Lambda \in [57.66, 57.665]$). In the stable areas, the sole use of a previously estimated underlying state should turn out to be very useful. In the less stable areas, it is not clear how much such an estimate would help.

Figure 3.2 implies that appending temporal representations to each other should

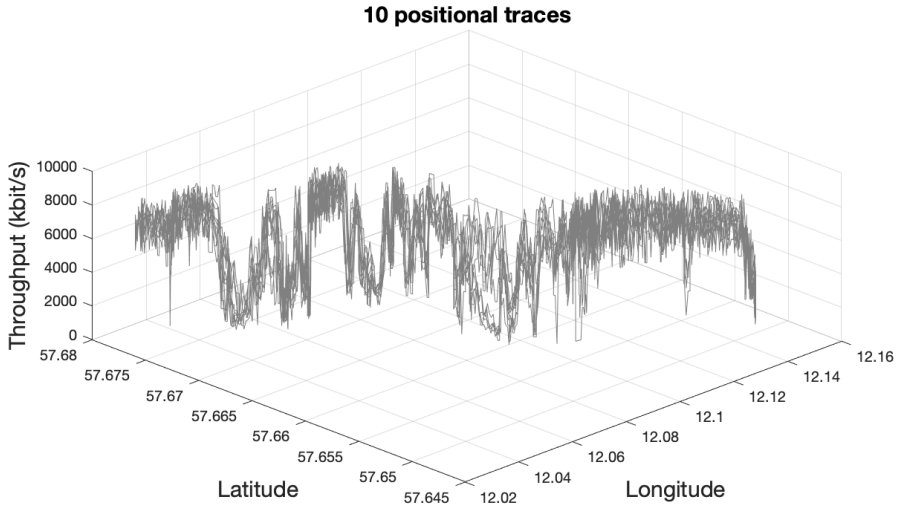


Figure 3.2: The total positional representation of the 10 trips, appended to each other.

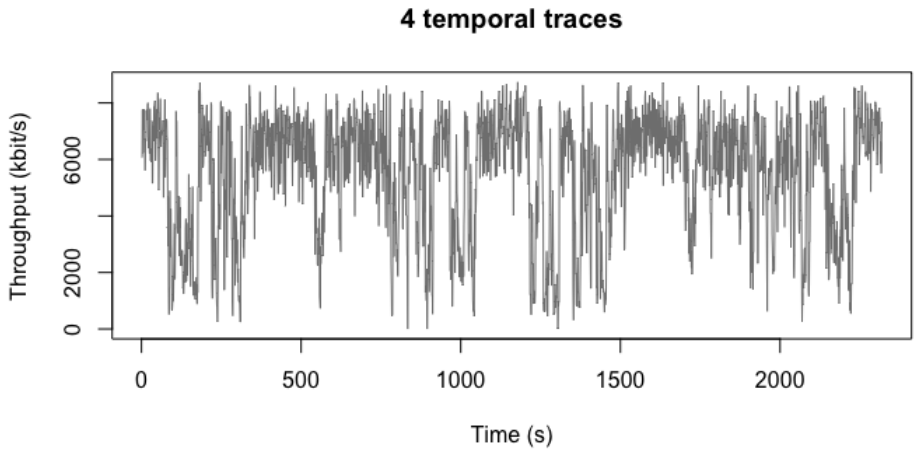


Figure 3.3: The temporal representation of the first 4 trips appended to each other.

result in a cyclic time series. Indeed, Figure 3.3 shows *cyclicity*, the presence of recurring patterns with irregular time intervals. To see that the time series is indeed *cyclic* and not *seasonal*, the 5 traces from Mölnlycke to Delsjömotet can be plotted on top of each other. This results in Figure 3.4. Since every trip is subject to unique happenstances, every trip will result in different travel times. Because of this, the total time series of all data traces appended to each other cannot be regarded as a pure seasonal one, but rather a cyclic one.

For the rest of the thesis, a particular data trace will be studied more than others; the trace of the 10^{th} trip. This is in order to stay consistent, but not overwhelm the text with plots of all different trips. (However, carrying out the same analyses on other trips is possible.) Looking at the temporal representation of the 10^{th} trip and its sample ACF and PACF gives some insight into the behaviour of these type of time series.

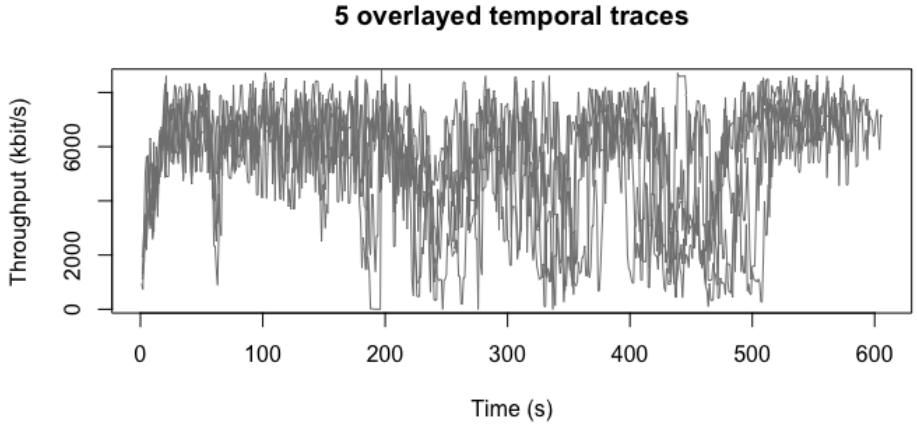


Figure 3.4: An overlay plot of the 5 temporal representations given by driving from Mölnlycke to Delsjömotet.

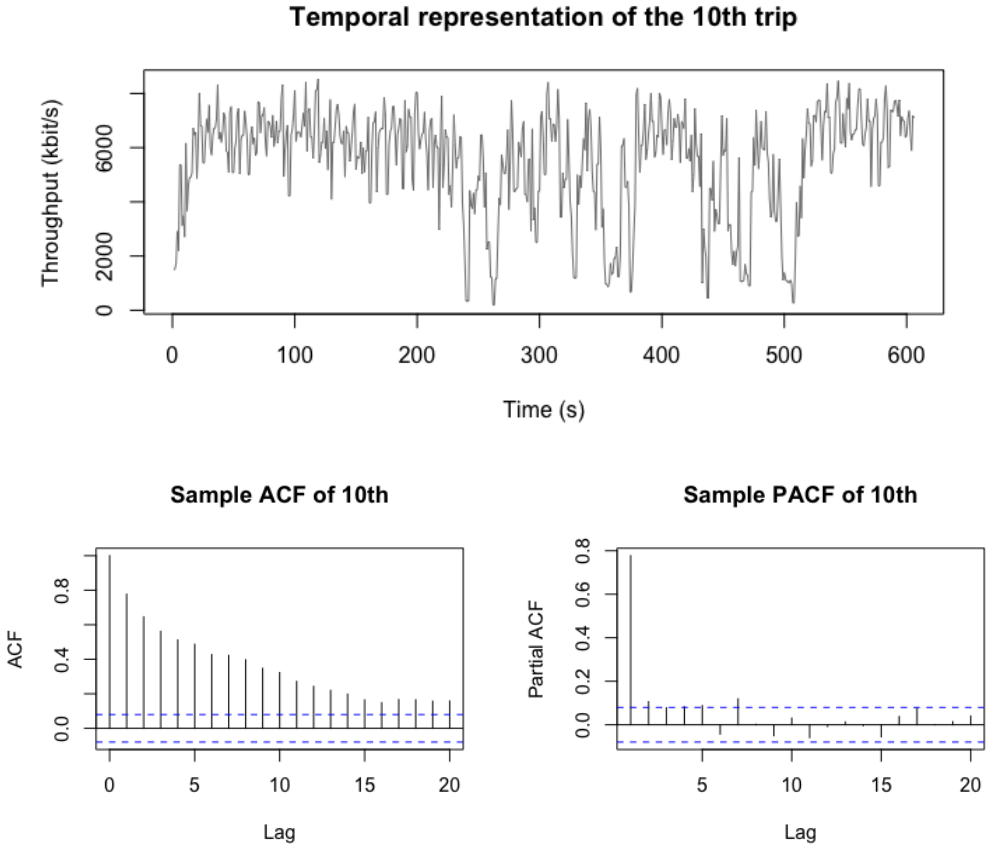


Figure 3.5: The temporal representation and sample ACF and PACF of the 10^{th} trip.

In Figure 3.5, the ACF shows the nonstationarity of the time series, while the PACF seems to indicate that it is almost a random walk, except for some subtle partial autocorrelation for small lags.

A 1-differencing leads to Figure 3.6, which indicates that an ARIMA(0,1,1) or ARIMA(0,1,2) model could fit. However, letting these serve as models for the time series in figure 3.5 results in rejections by the Ljung-Box test and they are thus not enough. Using AICc instead, the model turns out to have to be ARIMA(3,1,5), a fairly complex model, in order not to get rejected. This example shows how problematic manual order selection can be, but also how little the correlograms say about the order of ARMA models. Standard ARIMAs like these will be considered as alternative algorithms in later chapters, for the sake of comparing how much better forecasts become by using the cyclicity.

Estimating the underlying state in Figure 3.2 is a possible way to utilize the cyclicity. This essentially means combining the temporal and positional representations of the data traces in some way. How to do this is the subject of the rest of this chapter.

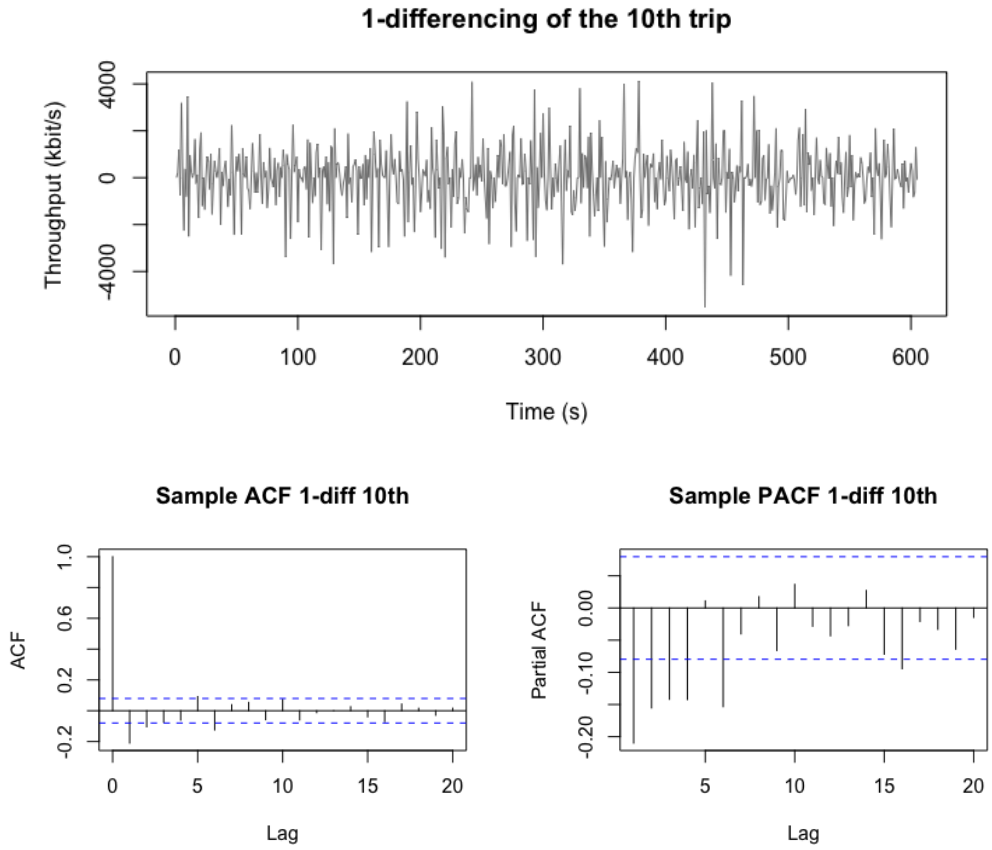


Figure 3.6: The 1-differencing of the 10$^{\text{th}}$ trip and its correlograms.

3.3 On the calculation of bandwidth maps

The underlying state present in Figure 3.2 is what the authors of [9] and [10] have tried to attain, i.e., it is what would constitute a *bandwidth map* of the route. In [9], the system queried for a predicted ABW once every 100 meters. The prediction was given by the mean ABW of all historical data points within a 100 meter radius. There are major problems with this strategy:

- The averaging over a set area results in the assumption that all positions in that area have the same underlying state, which is not necessarily the case.
- It uses an arbitrary choice of what will henceforth be called *window*, defining the size of the area and thus which historical points to use.
- This window is constant in size and not dependent on, for example, travelling speed.
- All points are weighted equally during the averaging.

These problems result in a bandwidth map that is too *smooth* and misses out on potential prediction possibilities, as can be seen in the comparison in Figure 3.8.

In [10], a different approach was taken when constructing a bandwidth map. This

approach could be regarded as even more arbitrary and also less automatic. The authors manually divided a route up into 500 meter segments beforehand. They then logged throughput levels in these segments and let every segment's bandwidth map level be defined by the average of the data points logged in it. In a sense, it is equivalent to the previously mentioned strategy, but with 500 meter query intervals using 250 meter radius windows. However, the radius was actually never discussed in this paper, leaving out the discussion on how uncertain GPS measurements should be dealt with in a real-time system implementing this method.

Motivated by the shortcomings of the existing methods for calculating bandwidth maps, a new method is introduced in this thesis. Using GPR on throughput measurements paired with their distance along the route from a specified starting point (say Delsjömotet), it is possible to attain a differentiable bandwidth map along the entire route. This allows for a minimal amount of arbitrary parameter choices and quick convergence to a stable bandwidth map. The rest of this section is dedicated to the necessary preprocessing steps for this method.

There are a few problems with implementing GPR on the data used in this thesis. The GPR of consideration is a two-dimensional one (because higher dimensions result in infeasible computational complexities). This means the data points need to be represented in a two-dimensional space. However, the positional representation of the data is three-dimensional. Therefore, a suitable transformation between the two is needed.

The transformation needs to consist of several steps. The first step is calculating an approximation of the route in the GPS-plane. This is because the exact GPS coordinates of the route will not necessarily be known. The second step is to remove the noise within the GPS coordinates of the data points. This can be done by projecting every data point's GPS coordinates onto the approximated route. The third and last step is to make the approximated route represent the x-axis of a two-dimensional space.

In this thesis, the first step is done through a linear interpolation of the first trip's

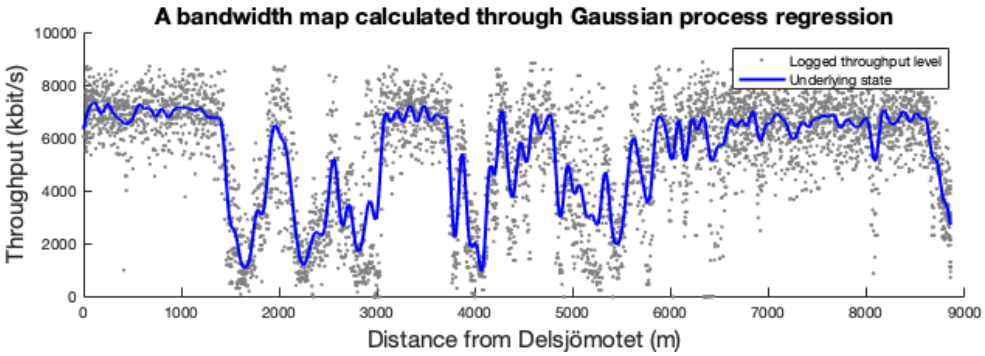


Figure 3.7: Gaussian process regression on throughput measurements paired with distances along the route.

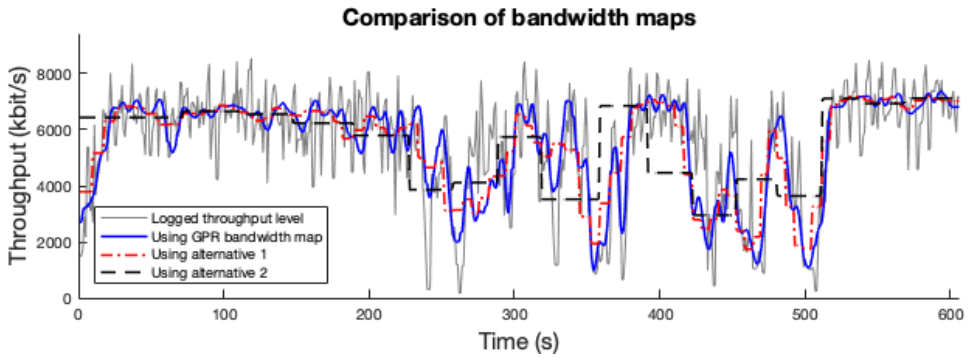


Figure 3.8: The 10^{th} trip in its temporal representation and the predictions given by various bandwidth maps.

positions. The second step is done through an orthogonal projection onto the piecewise linear curve given by the first step. The third step is achieved by letting the x-axis of the two-dimensional space be represented by the distance along the piecewise linear curve. Preprocessing the data points in this manner results in the points seen in Figure 3.7. Fitting a GPR with a Gaussian kernel to this data results in the line seen in the same plot.

Points from the first 9 trips were used in Figure 3.7. This bandwidth map can then be put alongside the temporal representation of the 10^{th} trip. This is done by applying the second step of the preprocessing procedure to every data point of the 10^{th} trip. Every projected position implies a distance along the route, which in turn implies a value of the bandwidth map. The result is seen in Figure 3.8, together with the methods used in other articles (alternative 1 is the one used in [9], alternative 2 the one in [10]).

3.4 The PARIMA model

This chapter ends with the introduction of a new type of time series model, the *Positional ARIMA (PARIMA)* model. It is an exogenous time series model, meaning it depends on some external, independent time series.

Definition. The series $\{X_t\}$ is a PARIMA(p, d, q) process if it has the representation

$$\phi(L, p)\Delta^d \left[X_t - \psi(\Phi_t, \Lambda_t)\right] = \theta(L, q)Z_t, \quad \{Z_t\} \sim \text{WN}(0, \sigma^2), \tag{3.1}$$

where $\{\Phi_t\}$ and $\{\Lambda_t\}$ are exogenous univariate time series and $\psi : \mathbb{R}^2 \to \mathbb{R}$. □

The subtraction carried out inside the brackets in (3.1) will be referred to as *positional differencing*. The definition of ψ allows for the use of some coordinate-based look-up-procedure, like the second step of the preprocessing based on orthogonal projection, described in the previous section.

This definition makes model specification through positional differencing and ARIMA

modeling on the resulting residuals possible. Essentially, such a model uses the bandwidth map to reduce the complexity of the temporal representation of the data, which then only portrays the deviation from the underlying state over time. Fitting an ARIMA model to this deviation is a way to refine the predictions and deal with less stable areas.

PARIMA(p, d, q) modeling on the 10^{th} trip starts with the positional differencing of the temporal representation. If ψ is the orthogonal projection procedure, the residuals from the positional differencing are shown in Figure 3.9.

The correlograms of Figure 3.9 resemble those of an AR(1) process (Figure 2.4). Indeed, fitting such a model to the data produces seemingly uncorrelated normally distributed residuals (Figure 3.10). However, this is rejected by a Ljung-Box test. Using AICc leads to PARIMA$(2, 1, 4)$, which is not rejected. The positional differencing has thus reduced the complexity necessary to describe the data by 25%, from $p + q = 8$ to $p + q = 6$. Moreover, it reduced the *Root Mean Squared Error (RMSE)* of the prediction (more on this in Chapter 5).

The PARIMA$(1, 0, 0)$ results in an RMSE of 1084.917 kbit/s, an improvement on

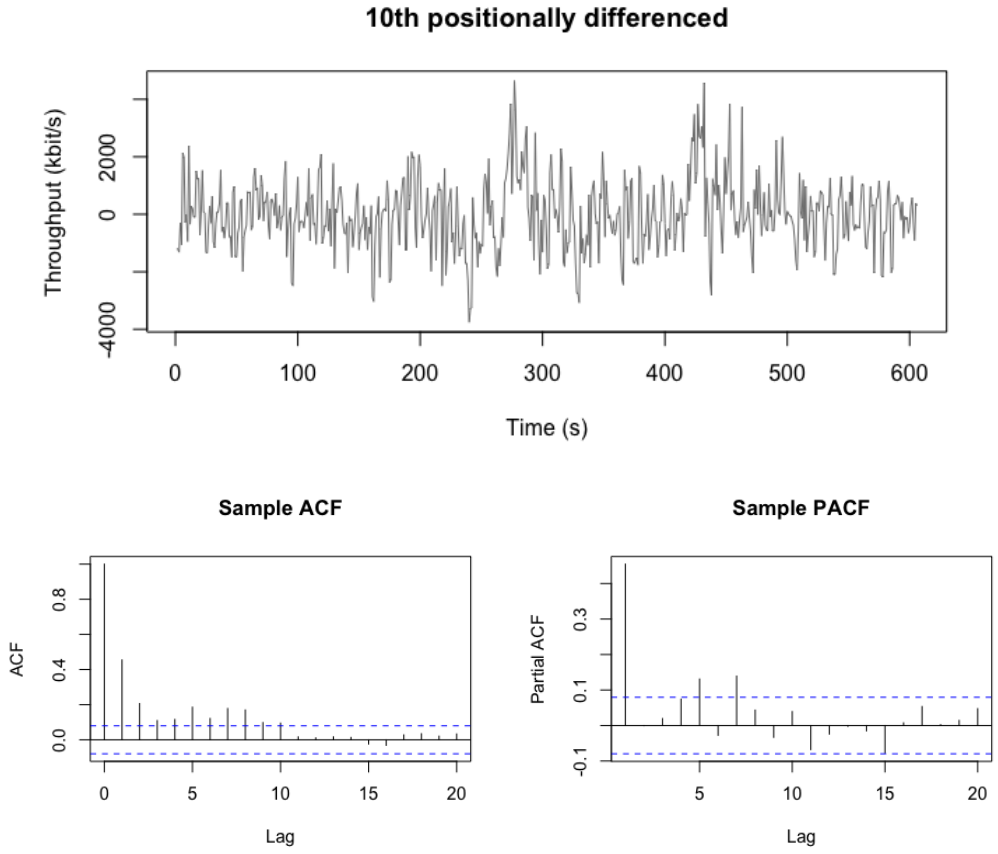


Figure 3.9: The temporal representation and sample ACF and PACF of the 10^{th} trip.

the predictions given by only the bandwidth map, which has an RMSE of 1218.402 kbit/s. PARIMA$(2, 1, 4)$ results in a RMSE of 1076.947 kbit/s. This small difference makes PARIMA$(1, 0, 0)$ preferable for a lightweight predictive real-time algorithm.

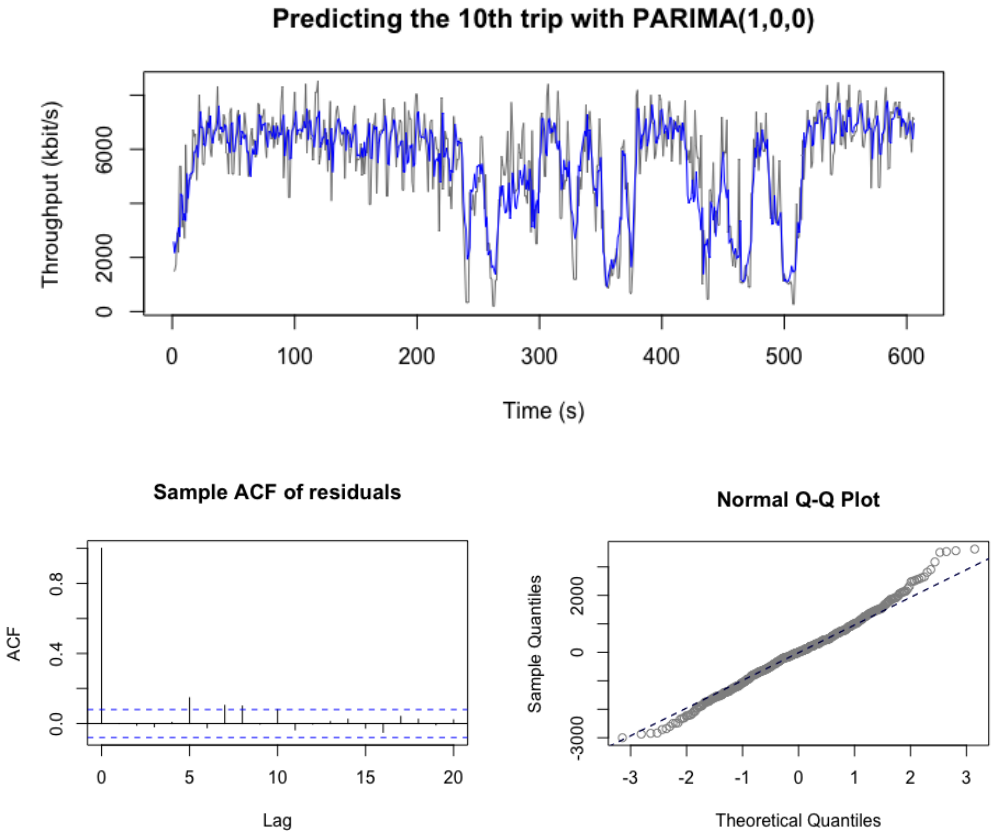


Figure 3.10: Result of 1-step-ahead predicting the temporal representation of the 10$^{\text{th}}$ trip with PARIMA(1,0,0) and the sample ACF and Q-Q plot of the residuals.

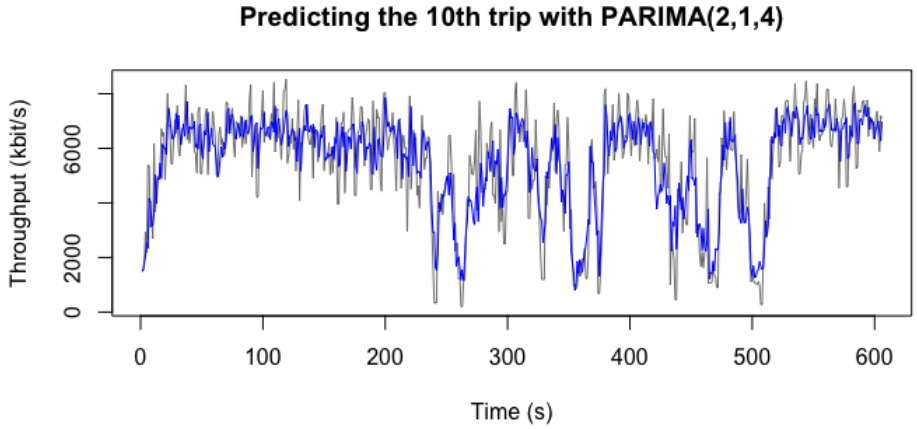


Figure 3.11: Result of 1-step-ahead predicting the temporal representation of the 10$^{\text{th}}$ trip with PARIMA(2,1,4).

4

Specifications of Algorithms

In this chapter, different algorithms for computing the lower boundary $B(t)$, described in Section 1.3, are specified. First, a reactive algorithm is introduced, in order to have a basic implementation to compare the other algorithms to. This is followed by a discussion regarding the problems faced by position-based real-time algorithms. The chapter concludes with the introduction of a PARIMA-based algorithm and alternative ones based on only ARIMA or only bandwidth maps. The performances of all of these algorithms will be compared in Chapter 5.

4.1 A reactive algorithm

Reactive algorithms are common in networking. When applied to automatic bitrate adjustment, they are frequently based on controllers of different kinds, which often use all of the network measures mentioned in Section 1.2. Examples over TCP are *TCP Friendly Rate Control (TFRC)*, used in [10], and *Bottleneck Bandwidth and Round-trip propagation time (BBR)*, a more modern alternative to TFRC [40]. In [41], a control algorithm for UDP called C^{3}G, mimicking the behaviour of BBR, was also implemented successfully. Moreover, Einride's current system uses a type of multivariate controller.

Since the data used in this thesis lacks delay and packet loss information, the reactive algorithm has to only work with throughput. Because of this, the algorithm was made to mimic the behaviour of C^{3}G's rate control, using only throughput (Figure 4.1). It is a modified discrete-time PID controller, where the integrated term has been changed for the current value and a few extra features have been introduced.

Algorithm. If the boundary specified by the **Reactive Algorithm (RA)** is referred to as $B(t)$ and X_t is the ABW at time t, with r, S, K_p, K_i, and K_d as free parameters, the algorithm behaves as follows:

1. Initialization through $B(1) = X_1$, assuming X_1 is known. The value of $B(t)$ is bounded by the minimum allowance b.
2. For every time step t, one of the following actions is carried out ($B(t)$ is still bounded by the minimum allowance b):
   - If $B(t) > X_t$, then $B(t+1) = X_t - r[B(t) - X_t]$, meaning r specifies rest in relation to overload.
   - If $B(t) \leq X_t$, time t is regarded as *successful* and $B(t+1) = B(t)$.

However, if S consecutively successful time steps have been experienced, the update is instead

$$B(t+1) = K_p[X_t - B(t)] + K_i B(t) + K_d[B(t) - B(t-1)]. \quad (4.1)$$

Clearly, the above needs a counter for consecutively successful steps. The alternative update exists in order for the algorithm to be able to probe for higher bitrates. □

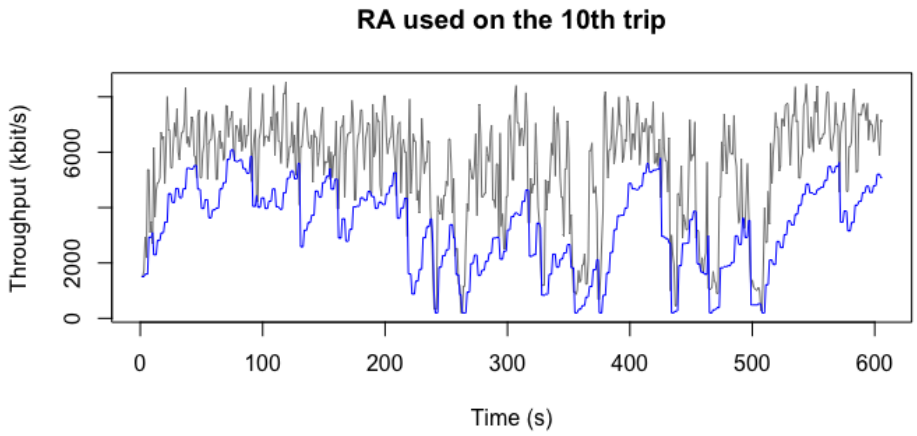


Figure 4.1: RA with $r = 1$, $S = 3$, $K_p = 0.1$, $K_i = 1$, and $K_d = 0.3$. Minimum allowance arbitrarily set to 200 kbit/s.

4.2 Problems for position-based algorithms

When utilizing any position-based look-up procedure in a real-time algorithm, the problem of predicting upcoming positions arises. Given a position (Φ_t, Λ_t) and assuming a time granularity of 1 second between data points, the estimate $(\hat{\Phi}_{t+1}, \hat{\Lambda}_{t+1})$ can easily be calculated with the velocity v_t at time t. If the velocity is not available, it can be approximated by the vector from $(\Phi_{t-1}, \Lambda_{t-1})$ to (Φ_t, Λ_t). However, as soon as $(\hat{\Phi}_{t+h}, \hat{\Lambda}_{t+h})$, $h > 1$, is to be considered, the potential change in direction and speed has to be dealt with.

In this thesis, the approximation of the route, described in Section 3.3, is used to predict upcoming positions. The coordinates (Φ_t, Λ_t) can be orthogonally projected onto the route and the resulting coordinates can be used to predict upcoming positions along the route. This can be done using speed estimates, assuming the driving direction is known.

If current speed is an available variable that can be logged in the system, there is the possibility of building a *speed map* through GPR as well (Figure 4.2). This method has multiple positive aspects to it. Besides the lack of tuning parameters, the actual speed s_t at time t can be utilized. This can be done by calculating the ratio between the actual speed at time t and the corresponding speed map value,

i.e., $s_t/\psi_s(\Phi_t, \Lambda_t)$, and use this to adjust the speed map so that it fits s_t. This will serve all sorts of trips quite well, even the ones subjected to anomalistic speeds. Examples of such scenarios are ending up behind a tractor or in a traffic jam. It can be improved by incorporating acceleration. This would help in a situation when a traffic jam suddenly clears up and the vehicle returns to the normal speed for the area, for example. However, an improvement like this is not studied in this thesis.

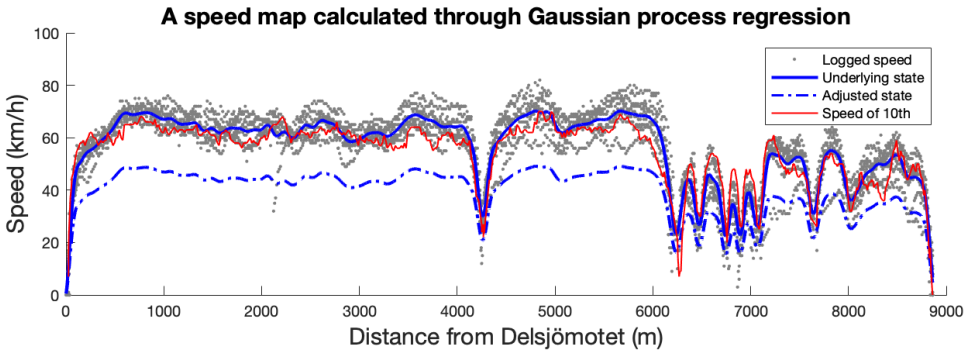


Figure 4.2: A speed map calculated through GPR. An example of an adjusted speed map is showcased, where $s_t = 0.7\psi_s(\Phi_t, \Lambda_t)$. The plot also includes the speed trace of the 10^{th} trip.

4.3 A PARIMA-based algorithm

The following algorithm is based on PARIMA modeling and is the algorithm proposed in this thesis in order to solve the problem described in Section 1.1.

Algorithm. With $B(t)$ and X_t defined as for RA, the **PARIMA Algorithm (PA)** is an algorithm with two *stages*. The first stage is *preparation*, which consists of three steps:

1. Calculation of bandwidth map through GPR.
2. Fitting of PARIMA(1,0,0) model, using the bandwidth map, to the last logged trip.
3. Calculation of speed map through GPR.

The second stage is *real-time prediction*, during which the following is carried out once for every h time steps:

1. Estimation of h upcoming positions through speed map and current position.
2. Prediction of X_{t+i}, $i = 1, ..., h$, using the PARIMA model attained during the preparation stage applied to the estimated upcoming positions.
3. Calculation of $B(t+i)$, $i = 1, ..., h$. This is done by subtracting a set amount of standard deviations, given by the bandwidth map, from every prediction X_{t+i}, $i = 1, ..., h$. This offset is related to the expected rate of failure. Moreover, $B(t)$ is always bounded by the minimum b. □

Ideally, the preparation stage should be rerun after every new trip. Moreover, it is entirely possible that weekday, time of day, month or season, and other aspects affect the bandwidth map (although, articles like [10] have shown indications that such differences are not significant). Therefore, an idea to improve the algorithm over time is to classify traces due to aspects like these and build multiple bandwidth maps. The real-time prediction stage would then only use the bandwidth map of its classification. Due to data limitations, this is not investigated in this thesis.

4.4 Alternative algorithms

In this short section, alternatives to the previously mentioned algorithms will be introduced. These will serve as comparisons in order to investigate how much better PA performs and determine if it is actually useful or if a simpler algorithm is to be preferred.

Algorithm. The **ARIMA Algorithm (AA)** only utilizes an ARIMA model fitted to the temporal representation of the last logged trip. This is then used to predict the throughput of a new trip in real-time. The boundary $B(t)$ is given by a set amount of standard deviations due to the estimated variance of the ARIMA model, similar to the method used in PA, again bounded by the minimum allowance b. □

Algorithm. The **Map Algorithm (MA)** is equivalent to PA with a PARIMA(0,0,0) model, i.e. it only uses the bandwidth map for its prediction step. □

Algorithm. The **Norwegian Algorithm (NA)** is the algorithm proposed in [9]. It uses the first alternative bandwidth map seen in Figure 3.8 to predict the bandwidth. In a similar fashion as this bandwidth map is created, through averaging, a variance-based map is also created. The boundary $B(t)$ is then defined by subtraction of a set amount of standard deviations, which are due to this variance-based map. Of course, $B(t)$ is yet again bounded by the minimum allowance b at all times. □

The amount of standard deviations subtracted is directly related to the expected rate of failure mentioned in Section 1.3. Section 5.3 discusses how to attain a proper amount of standard deviations for each of PA, AA, MA, and NA in order to compare these. Therefore, this is not discussed here.

5
Algorithm Performance

This chapter presents the performance results. Many of them have already been shown in plots throughout the text, but this chapter aims to compare the different algorithms based on some predefined performance metrics. These are presented in a table accompanied by plots. Moreover, this chapter discusses the convergence time of bandwidth maps, i.e., how many trips are needed until a reasonable underlying state has been attained. The chapter also includes discussions about how free parameters of the algorithms are optimized and the time complexities of the algorithms. Throughout all of this chapter, the target failure rate r will be 5% and the minimum allowed value b for the lower boundary $B(t)$, mentioned in Section 1.3, will be 200 kbit/s, for all algorithms.

5.1 Performance metrics

To make it easy to compare the algorithms, certain performance metrics need to be defined. According to (1.1), (1.2), and (1.3) of Section 1.3, the aim is to optimize the sum of the lower boundary $B(t)$. However, the magnitude of $B(t)$ gives very little actual insight into the performance of an algorithm after it has been used. All algorithms of Chapter 4 use the lower bound of some confidence interval around a prediction of ABW to define $B(t)$. This confidence interval is given by an *a priori* scaling factor, which will be discussed in further detail in Section 5.3. Hence, $B(t)$ is mainly useful in the sense that it is a common measurement for all algorithms, directly related to the individual scaling factor of each algorithm. Therefore, an *a posteriori* measure that is easily appreciated is introduced.

Definition. The a posteriori measurement **Mean Channel Usage (MCU)** is defined as the average ratio between the boundary $B(t)$ and the throughput T_t of successful times $t \in N := \{1, 2, ..., n\}$, for a trip of n time steps, i.e.,

$$\text{MCU} = \frac{1}{|\mathcal{S}|} \sum_{t \in \mathcal{S}} \frac{B(t)}{T_t}, \quad \mathcal{S} := \{t \in N : T_t \geq B(t)\}.$$

□

The reason for using only successful time steps is due to the fact that unsuccessful ones would increase this metric, which would paint a false picture of success, rewarding failure.

Other than MCU, $\Sigma B = \sum_{t \in N} B(t)$ will also be considered, as well as the RMSE of the prediction without the confidence interval offset. The latter is useful to know in cases where MCU does not differ significantly between two algorithms. It portrays if either prediction in itself would be more useful than the other in some alternative scheme that would only use the prediction (such a scheme is, for example, used in [10]).

5.2 Bandwidth map convergence

Studying the amount of change to the bandwidth map over time, given more and more traces, says something about the maximum amount of traces that can be used until another makes little difference. This is interesting because if the bandwidth map indeed turns out to be different for different times of the day or season, etc., as discussed in 4.3, or change over time, then the bandwidth map is not static and needs to be updated. Since using a lot of traces makes the GPR computation very heavy, this measure can serve as a specification for the amount of traces that should be utilized in the update.

The difference between successive bandwidth maps over amount of trips can be seen in Figure 5.1. It seems to follow an exponential decay. The result of fitting such a line to the collapsed mean of the surface plot on the left is seen in the plot to the right. The function representing this line has a derivative of approximately -9.2 at Trip $= 11$. This means that adding another data trace to a set of already 10 traces would on average change the bandwidth map with less than 10 kbit/s in each position.

Another way to study the convergence of the bandwidth map is to study the performance of the Map Algorithm. Studying its performance on the 10^{th} trip, given more and more previous traces, results in Figure 5.2. All performance metrics show clear trends, seemingly related to the one in Figure 5.1. This makes sense, since including

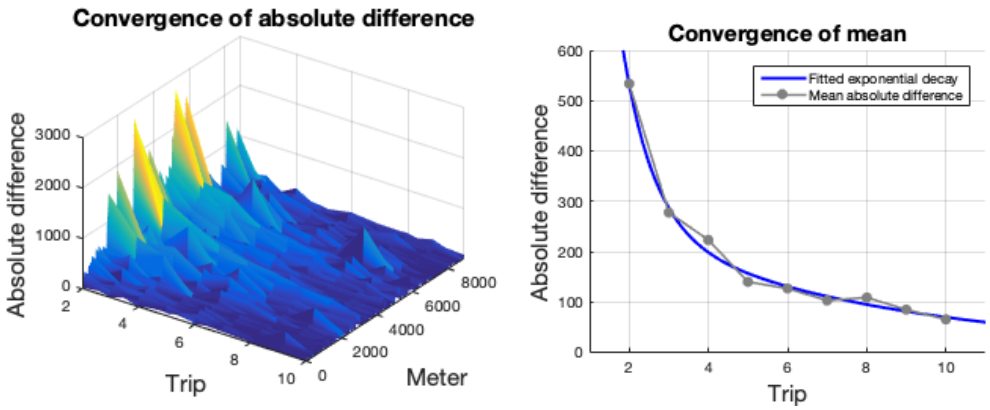


Figure 5.1: The convergence of the whole bandwidth map and the convergence of the mean along the Meter-axis.

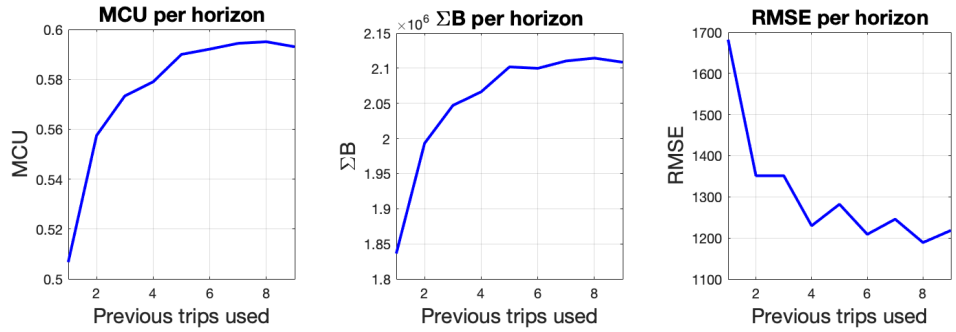


Figure 5.2: Performance metrics of the Map Algorithm on the 10^{th} trip for an increasing horizon of historic data to use, showing only the 9^{th} trip being used for the bandwidth map, then the 9^{th} and 8^{th}, and so on.

more and more trips will help the performance less and less due to the little difference it does to the map. However, this method also shows an interesting feature of the route. The periodicity on top of the trend, most clearly seen in the RMSE, is due to the fact that a drive from Deljömotet to Mölnlycke will be subjected to a slightly different bandwidth map than a trip from Mölnlycke to Delsjömotet. This can be seen in Figure 5.3.

Not surprisingly, the performance becomes better using only data from trips of the same direction! For the 10^{th} trip, letting the target failure rate be 5%, the MCU of the Map Algorithm turned out to be 59.3% when using all previous 9 trips, while it

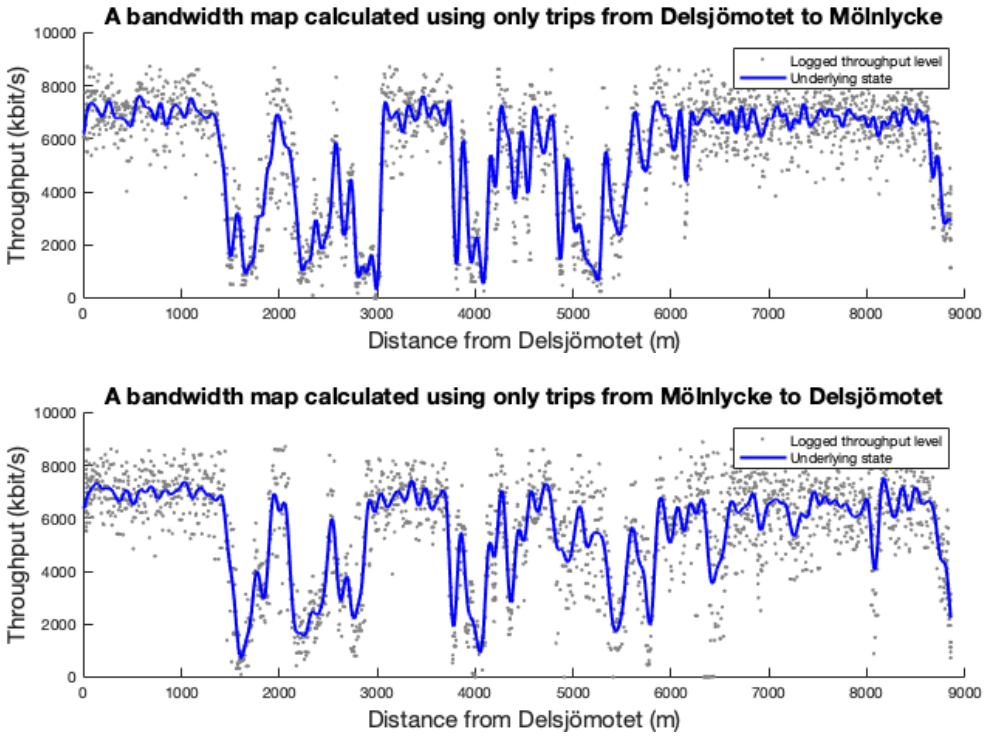


Figure 5.3: A comparison of bandwidth maps given different directions.

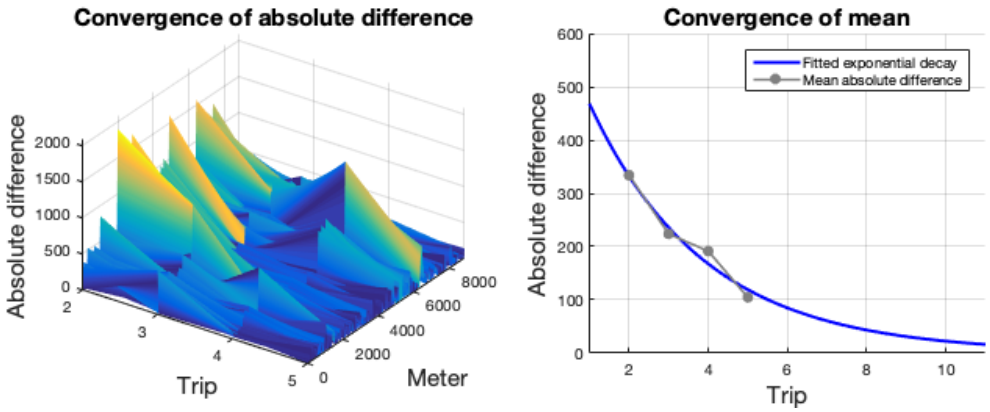


Figure 5.4: The same type of convergence plots as those of Figure 5.1, using only trips from Mölnlycke to Delsjömotet.

was 62.8% when using only the 4 trips of the same direction.

This also has an effect on the convergence plots. In Figure 5.4, it is clear that the severe high points, which were in the unstable areas, are now closer to the mean. Moreover, fitting an exponential decay to this data produces a function with derivative -5.15 at Trip $= 11$, i.e., almost half that of the previous exponential decay in Figure 5.1. The performance metrics over time of this method is presented in Figure 5.5.

According to Figures 5.4 and 5.5, there is still much to gain from additional trips. Thus, to expect an even better performance with more data traces from trips going in the same direction is completely reasonable, but using more than 10 is unnecessary. Einride and other implementers of algorithms based on GPR calculated bandwidth maps are therefore hereby advised to apply a common-direction look-back method of a horizon of about 10 traces. This can be easily updated during every preparation stage, as this amount of data should be manageable for GPR in a reasonable time frame. They are also advised to study the potential usefulness of various bandwidth maps for different times, as mentioned in Section 4.3.

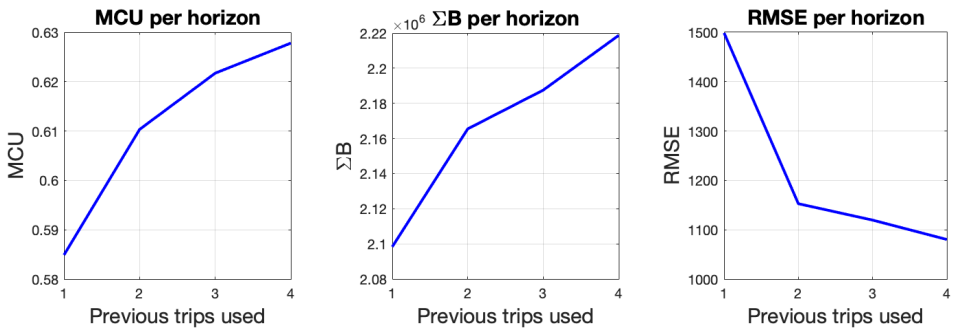


Figure 5.5: Performance metrics of the Map Algorithm through the same analysis as in Figure 5.2, but with only the trips from Mölnlycke to Delsjömotet.

5.3 Optimization and complexity aspects

Before the upcoming section of algorithm performance comparisons, it is important to discuss the ways the various algorithms are tuned and optimized in order to be comparable. This section discusses the scaling factor of $B(t)$, the complexity of the PARIMA Algorithm, and the optimization of ϕ in the PARIMA(1,0,0) model.

As stated in Chapter 4, all of the algorithms, except the Reactive Algorithm, use a lowered version of some prediction as their definition for the lower boundary $B(t)$. This is done through finding a suitable scaling factor α to the standard deviation measure acquired by each algorithm. Assuming an *up-and-running system* with some set amount of former traces used in every bandwidth map for each trip, the *failure rates*, $\{\mathrm{F}_\tau\}_{\tau=1}^m$, of previous trips is of interest. The definition of α over time is equivalent to the minimization of the RMSE between $\{\mathrm{F}_\tau\}_{\tau=1}^m$ and the *acceptable rate of failure*, r, mentioned in Section 1.3. It is entirely possible that, for a constant α, $\{\mathrm{F}_\tau\}_{\tau=1}^m$ would not be white noise. Then the best predictor for the *expected failure rate*, $\hat{\mathrm{F}} = \epsilon/n$, of an upcoming trip $\tau = m+1$, would not necessarily be r. However, this was not studied in this thesis, due to the lack of data. Instead, in the comparisons in the next section, an a posteriori α_{m+1}, tuned to minimize $|\mathrm{F}_{m+1} - r|$, will be used. This is similar to assuming an up-and-running system and white noise behaviour of $\{\mathrm{F}_\tau\}_{\tau=1}^m$ for a constant α. The tuning can easily be done by a simple decreasing search over a reasonable interval, stopping when $|\mathrm{F}_{m+1} - r|$ starts increasing, to attain an as big α as possible. (Since α is actually some positive and real-valued amount of standard deviations, a reasonable interval could be $[0, 4]$.)

The complexity of the PARIMA Algorithm can be divided into the algorithm's two stages. The preparation stage is subjected to GPR, which runs in $O(N^3)$, where N is the number of data points used in the training [37]. It also includes the specification of ϕ, but this is a straight forward computation, as will be shown below. The complexity of the prediction stage depends on the implementation of the noise correction of the GPS coordinates. The implementation that was used in the upcoming section (the orthogonal projection onto a linear interpolation of the 1st trip's GPS coordinates) runs in $O(N_1)$ for every such orthogonal projection, with N_1 being the amount of data points of the 1st trip.

The calculation of ϕ during the fitting of the AR(1) model, to the positionally differenced temporal representation of the latest trip m, is attained by the sample autocorrelation of this time series [28]. This is because $\phi = \rho(1)$ for an AR(1) process, meaning the best estimator $\hat{\phi}$ given by the Yule-Walker equations is $\hat{\rho}(1)$. This value, as seen in Section 2.1, can be calculated in $O(N_m)$.

5.4 Performance comparisons

In this section, the influence on MCU of the lag h will be analyzed for the Map Algorithm. Then performance metrics for all algorithms will be shown and discussed.

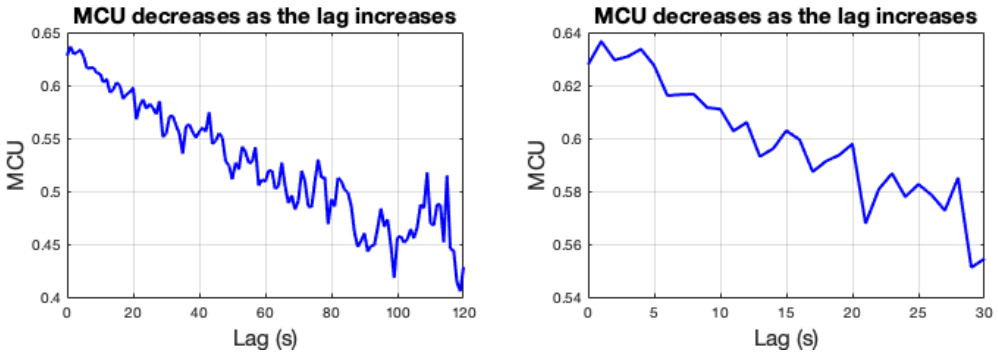


Figure 5.6: MCU of the Map Algorithm subjected to different lags, from 0 second to 2 minutes, where 0 seconds means no estimation needed, i.e. the upcoming position was assumed to be known. To the right is an enhancement of lags up to 30 seconds.

The lag h refers to how often a look-up in the bandwidth map is made, i.e., how often upcoming ABWs are predicted and bitrate levels are scheduled. The reason to only study the Map Algorithm is because the PARIMA Algorithm will converge quickly to the Map Algorithm if more than a lag of 1 second is studied, because the difference between the two is the AR(1) model, for which predictions of lags greater than 1 quickly converge to the mean [28]. The result of applying the speed map method (described in Section 4.2) to predict upcoming positions and predicting upcoming ABWs every h second is presented in Figure 5.6. A clear downwards trend is visible and the more seldom a look-up is made, the more random the performance seems to become. Note that the look-up can be made less often than every second, with 5 seconds having no detrimental effect to the performance. However, more seldom than twice every minute is not a good idea.

For the performance comparisons, the upcoming position is assumed to be known. This is, due to Figure 5.6, clearly a legitimate assumption to make, as several lags perform equally well. The performance metrics seen in Table 5.1 were measured for the various algorithms subjected to the 10^{th} trip.

	RA	AA	NA	MA	PA
MCU (1)	0.5528	0.5644	0.5014	0.6278	0.6426
ΣB (kbit/s)	1,850,000	2,007,000	1,855,000	2,219,000	2,273,000
RMSE (kbit/s)	-	1,189	1,459	1,080	1,021

Table 5.1: Performances of the different algorithms subjected to the 10^{th} trip and a target failure rate of 5%. The RMSE is due to the underlying prediction scheme before the creation of $B(t)$, which means the Reactive Algorithm lacks a value for this.

Comparing the performance of the Reactive Algorithm and the ARIMA Algorithm, the two reactive algorithms, shows that the latter is slightly better, although, this is a much more sophisticated algorithm. However, the usefulness of reactive algorithms is clear. A possible way to improve the performance of a proactive algorithm, like the PARIMA Algorithm, is to incorporate the Reactive Algorithm to work together with the PARIMA Algorithm. This could lead to better channel utilization.

Further, there are clear similarities between the Map Algorithm and the PARIMA Algorithm; they are almost equal in performance. It is however important to acknowledge that the step from the Map Algorithm to the PARIMA Algorithm is extremely small computationally: it is an $O(1)$ procedure, simply adding the knowledge of the throughput in the previous time step to the calculation. As such, the improvement of about 1.5% in MCU and a better prediction is enough to indicate the usefulness of the PARIMA Algorithm. All algorithms presented in this thesis show better performances than the Norwegian Algorithm.

To conclude this chapter, the result of the PARIMA Algorithm is presented in Figure 5.7. It seems to work well. However, its performance depends greatly on the magnitude of the variance of the noise in the time series. The difference in throughput from one measurement to the next, a second later, is frequently 3 Mbit/s or higher. This is very large considering the maximum recorded throughput is around 8 Mbit/s. During the 10^{th} trip, for example, this happened 4% of the time. As such, it is simply impossible to perform much better than what the PARIMA Algorithm already does. For example, letting the PARIMA Algorithm work with a PARIMA(2,1,4) model (Section 3.4) only increases MCU to 0.6459 and decreases RMSE to $1,008$ kbit/s. There are a few hiccups in its lower boundary, which are due to the bandwidth map. If these are eliminated by more data, the performance will be be even closer to optimal. As a closing remark, an important observation is that the RMSE of the ARIMA(3,1,5) model is about 18% larger than that of the PARIMA(2,1,4), which serves as a final justification for the usefulness of the latter.

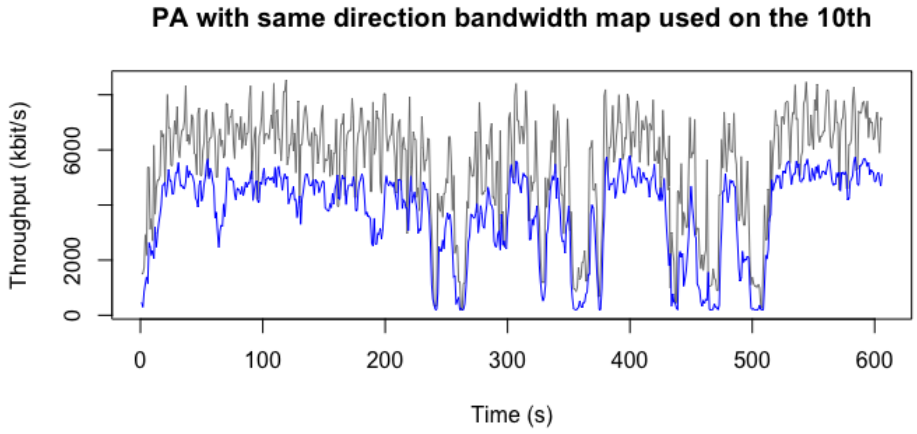


Figure 5.7: The plot corresponding to the performance metrics of the PARIMA Algorithm in Table 5.1.

6
Conclusions

In this concluding chapter, some remarks on the performance results illustrated in Chapter 5 are given. These include conclusions and ideas for improvements. This is followed by some closing words on possible future work that is less connected to the methods used in this thesis, but could be interesting for further research.

6.1 Result remarks

From the results in Chapter 5, it can be concluded that a fully working real-time algorithm for bitrate planning has been developed and presented in this thesis. Moreover, the discussion regarding the convergence of bandwidth maps showed the potential improvement of the performance of PA. Given more data, this performance can possibly tend towards MCUs of 70%.

Another way to improve the performance of the PARIMA Algorithm could be to implement a system for dealing with data traces of trip's during which the throughput behaved in nonrepresentative ways. For example, a large-scaled electrical outage during a trip would result in a drop in throughput, which would be logged in the trip's data trace. If this trace was included in the calculation of the bandwidth map, it would worsen the predictive performance of the algorithm for future trips. Excluding such traces through some statistical procedure during the preparation stage could be a way to improve the bandwidth map.

The amount of data is, as mentioned, very important. Not only could multiple bandwidth maps be created if more data was available, but the performance over time could be analyzed as a time series in its own right. This was briefly mentioned in Section 5.3. Doing this could greatly affect the accuracy of the scaling factor α needed to achieve a failure rate as close as possible to the target rate r.

Lastly, the performance of the PARIMA Algorithm could of course simply be improved by increasing the complexity of the underlying ARIMA model. In fact, although the improvement would not be significant (as shown in Section 5.4), the model fitting would only be done during the preparation stage, which already has a complexity of $O(N^3)$, and as such would not imply any complexity problems. The bigger problem with this improvement is the actual implementation of such a system, since automatic ARIMA modeling libraries are uncommon among programming languages.

The major limitation in this work is the small amount of data. It could be that more test data would show unforeseen problems. However, the general nature of the algorithm and the fact that it works on only a tiny amount of data seems makes it seem promising.

6.2 Future work

The first suggestion for future work is the combination of the PARIMA Algorithm and some reactive algorithm. In [10], this is the way the bandwidth map was utilized. This seems to have worked well and based on the performance of the Reactive Algorithm, the combination of the Reactive Algorithm and the PARIMA Algorithm or the Map Algorithm seems like an algorithm that could perform very well. For example, if the Map Algorithm is used with a lag of 10, RA can be put into action as soon as a new level is set. This can be combined with modifying the resting parameter r of the Reactive Algorithm to take into account the lower boundary $B(t)$ given by the Map Algorithm. Another way to use a reactive algorithm with the PARIMA Algorithm is to use the prediction of the PARIMA Algorithm as the throughput measure in some controller, e.g., the one used by Einride at the moment. This would make the controller proactive instead of reactive and could prove very useful.

The second suggestion is the utilization of historic ABW measurements to create *bandwidth surfaces*, similar to those created in geostatistics through Gaussian process regression to estimate changing levels of some measure over an area, given scattered historic measurements. Doing this would allow ABW to be predicted not only along a certain route, but for every point in a given area. Such surfaces would be useful in big open areas, like parks, where movement is not constrained, to predict upcoming ABW for mobile users.

Bibliography

[1] J. F. Kurose and K. W. Ross, *Computer networking: a top-down approach*. Boston: Pearson, seventh edition ed., 2017.

[2] D. Comer, *Internetworking with TCP/IP*, vol. 1. Pearson, 6 ed., 2013.

[3] D. Comer, *Computer networks and internets*. Upper Saddle River, N.J: Pearson/Prentice Hall, 5th ed ed., 2009.

[4] I. E. Richardson, "The H.264 Advanced Video Compression Standard, Second Edition," p. 349.

[5] X. Yin, A. Jindal, V. Sekar, and B. Sinopoli, "A Control-Theoretic Approach for Dynamic Adaptive Video Streaming over HTTP," in *Proceedings of the 2015 ACM Conference on Special Interest Group on Data Communication - SIGCOMM '15*, (London, United Kingdom), pp. 325–338, ACM Press, 2015.

[6] T. Mangla, N. Theera-Ampornpunt, M. Ammar, E. Zegura, and S. Bagchi, "Video through a crystal ball: effect of bandwidth prediction quality on adaptive streaming in mobile environments," in *Proceedings of the 8th International Workshop on Mobile Video - MoVid '16*, (Klagenfurt, Austria), pp. 1–6, ACM Press, 2016.

[7] N. Bui, M. Cesana, S. A. Hosseini, Q. Liao, I. Malanchini, and J. Widmer, "A Survey of Anticipatory Mobile Networking: Context-Based Classification, Prediction Methodologies, and Optimization Techniques," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1790–1821, 2017.

[8] H. Riiser, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Bitrate and video quality planning for mobile streaming scenarios using a GPS-based bandwidth lookup service," in *2011 IEEE International Conference on Multimedia and Expo*, (Barcelona, Spain), pp. 1–6, IEEE, July 2011.

[9] H. Riiser, T. Endestad, P. Vigmostad, C. Griwodz, and P. Halvorsen, "Video streaming using a location-based bandwidth-lookup service for bitrate planning," *ACM Transactions on Multimedia Computing, Communications, and Applications*, vol. 8, pp. 1–19, July 2012.

[10] J. Yao, S. S. Kanhere, and M. Hassan, "Improving QoS in High-Speed Mobility Using Bandwidth Maps," *IEEE Transactions on Mobile Computing*, vol. 11, pp. 603–617, Apr. 2012.

[11] J. v. d. Hooft, S. Petrangeli, T. Wauters, R. Huysegems, P. R. Alface, T. Bostoen, and F. D. Turck, "HTTP/2-Based Adaptive Streaming of HEVC Video Over 4g/LTE Networks," *IEEE Communications Letters*, vol. 20, pp. 2177–2180, Nov. 2016.

[12] S. Ahmad, R. Reinhagen, L. S. Muppirisetty, and H. Wymeersch, "Predictive resource allocation evaluation with real channel measurements," in *2017 IEEE*

International Conference on Communications (ICC), (Paris, France), pp. 1–5, IEEE, May 2017.

[13] D. Raca, J. J. Quinlan, A. H. Zahran, and C. J. Sreenan, "Beyond throughput: A 4g lte dataset with channel and context metrics," in *Proceedings of the 9th ACM Multimedia Systems Conference*, MMSys '18, (New York, NY, USA), pp. 460–465, ACM, 2018.

[14] M. Mathis, J. Semke, and J. Mahdavi, "The macroscopic behavior of the TCP congestion avoidance algorithm," *ACM SIGCOMM Computer Communication Review*, vol. 27, pp. 67–82, July 1997.

[15] B. Sikdar, S. Kalyanaraman, and K. S. Vastola, "Analytic models for the latency and steady-state throughput of TCP Tahoe, Reno, and SACK," *IEEE/ACM Transactions on Networking*, vol. 11, pp. 959–971, Dec. 2003.

[16] Yi Qiao, J. Skicewicz, and P. Dinda, "An empirical study of the multiscale predictability of network traffic," in *Proceedings. 13th IEEE International Symposium on High performance Distributed Computing, 2004.*, pp. 66–76, June 2004.

[17] M. Mirza, J. Sommers, P. Barford, and X. Zhu, "A Machine Learning Approach to TCP Throughput Prediction," *IEEE/ACM Transactions on Networking*, vol. 18, pp. 1026–1039, Aug. 2010.

[18] M. Mirza, K. Springborn, S. Banerjee, P. Barford, M. Blodgett, and X. Zhu, "On The Accuracy of TCP Throughput Prediction for Opportunistic Wireless Networks," in *2009 6th Annual IEEE Communications Society Conference on Sensor, Mesh and Ad Hoc Communications and Networks*, pp. 1–9, June 2009.

[19] B. Zhou, D. He, Z. Sun, and W. H. D. Ng, "Network Traffic Modeling and Prediction with ARIMA / GARCH," 2005.

[20] W. Yoo and A. Sim, "Time-Series Forecast Modeling on High-Bandwidth Network Measurements," *Journal of Grid Computing*, vol. 14, pp. 463–476, Sept. 2016.

[21] A. Biernacki, "Traffic prediction methods for quality improvement of adaptive video," *Multimedia Systems*, vol. 24, pp. 531–547, Oct. 2018.

[22] S. Gowrishankar and P. S. Satyanarayana, "A Time Series Modeling and Prediction of Wireless Network Traffic," *International Journal of Interactive Mobile Technologies (iJIM)*, vol. 3, pp. pp. 53–62, Jan. 2009.

[23] C. Katris and S. Daskalaki, "Dynamic Bandwidth Allocation for Video Traffic Using FARIMA-Based Forecasting Models," *Journal of Network and Systems Management*, vol. 27, pp. 39–65, Jan. 2019.

[24] Y. Liu and J. Y. B. Lee, "An Empirical Study of Throughput Prediction in Mobile Data Networks," in *2015 IEEE Global Communications Conference (GLOBECOM)*, (San Diego, CA, USA), pp. 1–6, IEEE, Dec. 2015.

[25] L. Cottrell, "pathChirp: Efficient Available Bandwidth Estimation for Network Paths," Tech. Rep. SLAC-PUB-9732, 813038, Apr. 2003.

[26] E. Bergfeldt, S. Ekelin, and J. M. Karlsson, "Real-time available-bandwidth estimation using filtering and change detection," *Computer Networks*, vol. 53, pp. 2617–2645, Oct. 2009.

[27] M. Li, Y.-L. Wu, and C.-R. Chang, "Available bandwidth estimation for the network paths with multiple tight links and bursty traffic," *Journal of Network and Computer Applications*, vol. 36, pp. 353–367, Jan. 2013.

[28] P. J. Brockwell and R. A. Davis, *Introduction to time series and forecasting*. Springer texts in statistics, New York: Springer, 2nd ed ed., 2002.

[29] A. Guntuboyina, "Spring 2013 Statistics 153 (Time Series) : Lecture Nineteen," 2013.

[30] J. Grandell, "Formulas and survey Time series analysis," p. 31.

[31] R. J. Hyndman and Y. Khandakar, "Automatic time series forecasting: the forecast package for R," *Journal of Statistical Software*, vol. 26, no. 3, pp. 1–22, 2008.

[32] R. Hyndman, G. Athanasopoulos, C. Bergmeir, G. Caceres, L. Chhay, M. O'Hara-Wild, F. Petropoulos, S. Razbash, E. Wang, and F. Yasmeen, *forecast: Forecasting functions for time series and linear models*, 2019. R package version 8.7.

[33] N. A. Heckert, J. J. Filliben, C. M. Croarkin, B. Hembree, W. F. Guthrie, P. Tobias, and J. Prinz, *Handbook 151: NIST/SEMATECH e-Handbook of Statistical Methods*. NIST Interagency/Internal Report (NISTIR), 2002.

[34] D. Kwiatkowski, P. C. Phillips, P. Schmidt, and Y. Shin, "Testing the null hypothesis of stationarity against the alternative of a unit root," *Journal of Econometrics*, vol. 54, pp. 159–178, Oct. 1992.

[35] T. S. Breusch and A. R. Pagan, "A Simple Test for Heteroscedasticity and Random Coefficient Variation," *Econometrica*, vol. 47, p. 1287, Sept. 1979.

[36] J. Durbin and S. J. Koopman, *Time series analysis by state space methods*. No. 38 in Oxford statistical science series, Oxford: Oxford University Press, 2nd ed ed., 2012.

[37] C. E. Rasmussen and C. K. I. Williams, *Gaussian processes for machine learning*. Adaptive computation and machine learning, Cambridge, Mass: MIT Press, 2006. OCLC: ocm61285753.

[38] K. P. Murphy, *Machine learning: a probabilistic perspective*. Adaptive computation and machine learning series, Cambridge, MA: MIT Press, 2012.

[39] Gyokov Solutions, "G-NetTrack Pro." `https://www.gyokovsolutions.com/G-NetTrack%20Android.html`.

[40] N. Cardwell, Y. Cheng, C. S. Gunn, S. H. Yeganeh, and V. Jacobson, "Bbr: Congestion-based congestion control," *Queue*, vol. 14, pp. 50:20–50:53, Oct. 2016.

[41] A. Alós, F. Morán, P. Carballeira, D. Berjón, and N. García, "Congestion Control for Cloud Gaming Over UDP Based on Round-Trip Video Latency," *IEEE Access*, vol. 7, pp. 78882–78897, 2019.