VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

MULTIKRITERIÁLNÍ GENETICKÉ ALGORITMY V PREDIKCI DOPRAVY

MULTI-OBJECTIVE GENETIC ALGORITHMS IN ROAD TRAFFIC PREDICTION

DISERTAČNÍ PRÁCE
PHD THESIS

AUTOR PRÁCE
AUTHOR

Ing. JIŘÍ PETRLÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Prof. Ing. LUKÁŠ SEKANINA, Ph.D.

BRNO 2016

Abstrakt

Porozumění chování silniční dopravy je klíčem pro její efektivní řízení a organizaci. Tato úloha se stává čím dál více důležitou s rostoucími požadavky na dopravu a počtem registrovaných vozidel. Informace o dopravní situaci je důležitá pro řidiče a osoby zodpovědné za její řízení. Naštěstí v posledních několika dekádách došlo k značnému rozvoji technologií pro monitorování dopravní situace. Stacionární senzory, jako jsou indukční smyčky, radary, kamery a infračervené senzory, mohou být nainstalovány na důležitých místech. Zde jsou schopny měřit různé mikroskopické a makroskopické dopravní veličiny. Bohužel mnohá měření obsahují nekorektní data, která není možné použít při dalším zpracování, například pro predikci dopravy a její inteligentní řízení. Tato nekorektní data mohou být způsobena poruchou zařízení nebo problémy při přenosu dat. Z tohoto důvodu je důležité navrhnout obecný framework, který je schopný doplnit chybějící data. Navíc by tento framework měl být také schopen poskytovat krátkodobou predikci budoucího stavu dopravy. Tato práce se především zabývá vybranými problémy v oblasti doplnění chybějících dopravních dat, predikcí dopravy v krátkém časovém horizontu a predikcí dojezdových dob. Navrhovaná řešení jsou založena na kombinaci současných metod strojového učení, například Support vector regression (SVR) a multikriteriálních evolučních algoritmů. SVR má mnoho meta-parametrů, které je nutné dobře nastavit tak, aby byla dosažena co nejkvalitnější predikce. Kvalita predikce SVR dále silně závisí na výběru vhodné množiny vstupních proměnných. V této práci používáme multiktriteriální optimalizaci pro optimalizaci SVR meta-parametrů a množiny vstupních proměnných. Multikriteriální optimalizace nám umožňuje získat mnoho Pareto nedominovaných řešení. Mezi těmito řešeními je možné dynamicky přepínat dle toho, jaká data jsou aktuálně k dispozici tak, aby bylo dosaženo maximální kvality predikce. Metody navržené v této práci jsou především vhodné pro prostředí s velkým množstvím chybějících hodnot v dopravních datech. Tyto metody jsme ověřili na reálných datech a porovnali jejich výsledky s metodami, které jsou v současné době používány. Navržené metody poskytují lepší výsledky než stávající metody, a to především ve scénářích, kde se vyskytuje mnoho chybějích hodnot v dopravních datech.

Klíčová slova

predikce dopravy, predikce dojezdových dob, multikriteriální optimalizace

Citace

Jiří Petrlík: Multi-objective genetic algorithms in road traffic prediction, Disertační práce, Brno, FIT VUT v Brně, 2016

Abstract

The understanding of the road traffic behavior is a key to effective traffic control, management and organization. This task is becoming more and more important with increasing traffic demands and the number of registered vehicles. The information about the current and future traffic situation is very important for drivers and traffic operators. Fortunately, there was a huge progress in technologies for traffic data acquisition in the last few decades. Stationary sensors, such as loop detectors, radars, cameras and infrared sensors can be installed on important locations of the roads and measure various microscopic and macroscopic traffic variables. However, some measurements can lead to an incorrect data which cannot further be used in the subsequent processing tasks such as traffic prediction or intelligent control. For example, this can be caused by equipment failures or data transmission problems. It is highly desirable to have a framework, which is capable of estimating the missing values in traffic data. It is also very important to provide a reliable short-time prediction of the traffic state. In this thesis, we focus on selected problems from this domain - the imputation of missing traffic data, short time traffic forecasting and travel times estimation. The proposed solution is based on combining the state-of-the art machine learning methods such as support vector regression (SVR) with the multi-objective evolutionary optimization. SVR has various meta-parameters which should be properly set in order to achieve the best performance. The performance also strongly depends on the selection of the input variables for SVR. We used the multi-objective optimization to find the proper settings of SVR meta-parameters and input variables. Using the multi-objective optimization, we obtained many different non-dominated solutions from Pareto front. These solutions can dynamically be switched according to the traffic data which are currently available, in order to maximize the quality of prediction. The proposed methods are specially designed for environments with many missing values in traffic data. We evaluated the proposed methods using real world data and compared them with the state of the art methods for the traffic data imputation and short term prediction such as the probabilistic principal component analysis and support vector regression optimized by a single objective optimization. The proposed methods provide better results than these state of the art methods especially in the cases where there are many missing values in the traffic data.

Keywords

road traffic forecasting, travel times estimation, multi-objective optimization

Bibliographic Citation

Jiří Petrlík: Multi-objective genetic algorithms in road traffic prediction, Ph.D. thesis, Brno, FIT Brno University of Technology, 2016

Multi-objective genetic algorithms in road traffic prediction

Prohlášení

Prohlašuji, že jsem tuto disertační práci vypracoval samostatně pod vedením svého školitele Prof. Ing. Lukáše Sekaniny Ph.D. a že jsem uvedl všechny literární prameny, ze kterých jsem v průběhu své práce čerpal.

.....
Jiří Petrlík
May 25, 2016

Poděkování

Rád bych na tomto místě poděkoval všem, kteří mi byli oporou při sepisování této práce. Chtěl bych poděkovat svému školiteli, panu Prof. Lukáši Sekaninovi za odborné rady a pomoc při sepisování práce. Rád bych poděkoval svým rodičům za podporu v průběhu studia. Děkuji také kolegům z UPSY, se kterými jsem měl možnost při svém studiu spolupracovat.

Contents

Chapter 1

Introduction

The understanding of road traffic behavior is a key to effective traffic control, management and organization. This task is becoming more and more important with increasing traffic demands and the number of registered vehicles. To the end of the year 2014, 6 775 877 vehicles were registered in the Czech Republic, which is 136 668 more registered vehicles than in 2013. The road traffic is essential for today economy and modern life. Unfortunately, it has also many negative effects such as traffic accidents or traffic pollution. According to Center of Traffic Research in Brno, the losses caused by traffic accidents were about 53 billions of Czech Crowns in the year 2013 [133].

These basic statistics show the importance of accurate information about the current traffic, which is available for drivers and for people responsible for road administration. To fulfill these demands, it is necessary to be able to measure the traffic flow variables. The first technology to measure the traffic flow was proposed in 1960s. This technology, called inductance loop, is based on the principle of induction. Inductance loops sustain to be a very important source of traffic data even today. Many other technologies based on various physical principles were proposed in past few decades, for example, traffic radars, magnetometers, infrared sensors and others. The modern computer vision algorithms enable us to detect the vehicles in video streams produced by traffic cameras. The visual data produced by these cameras contain much more information than other detectors, for example, number of vehicle license plate, vehicle type or size. However, the quality of provided information is very dependent on the quality of computer vision algorithms and actual weather conditions.

Traffic sensors like inductance loop detectors, radars, or traffic cameras are usually installed on one place are provide us with the information about the traffic on a given place. The other approach to measure the traffic flow is based on an equipment, which is installed into the chosen vehicles. This equipment is moving with a vehicle and provides the information about the vehicle position. We call the data obtained using this approach as a floating car data. However, with a development in communication and mobile devices, it is even not necessary to install a special device into the vehicle. Most of the drivers own a mobile phone and for mobile operators it is possible to track the position of these phones. Using the information about trajectories of phones, it is possible to get a very comprehensive information about the current state of the traffic. Unfortunately, it brings new ethical and security problems. Because of it, the obtained data are usually aggregated and anonymized in traffic systems.

Traffic sensors and floating cars produce a huge amount of data. It is necessary to transport and process these data. At first the data must be transported into the traffic centers. To enable this, many data protocols such as DatexII, Alert-C and others were

proposed [1]. Then the data, coming from different sources must be converted into the predefined common form. The process of this transformation is called the data fusion. After this, the data can be distributed to the drivers, or can be further processed using traffic modelling and traffic prediction algorithms.

The origins of traffic modeling and traffic flow theory dates backed to 1930s, when Bruce D. Greenshields performed observations of the traffic flow and postulated a simple traffic model [44]. Since then, various types of traffic models were proposed. These models typically differ in the aggregation level. The microscopic models work with each single vehicle and model its behavior. Contrary, the macroscopic models work with aggregated data and search for dependencies between the traffic variables. The traffic models also differ in mathematical base. They can be established on differential equations, cellular automata or other mathematical structures.

This thesis is mainly focused on traffic prediction based on machine learning algorithms. The machine learning algorithms are part of computer science, evolved from pattern matching and artificial intelligence. These algorithms are able to adapt their behavior according to used training data. This thesis primarily deals with supervised learning principle in which we have a set of training samples with known values of results. These data are called training data set. Using the training data, the machine learning algorithm is capable of finding dependencies between input values and desired outputs. The trained models are then evaluated using another data and used for prediction. In many previous works, it was shown that the machine learning methods are very successful in short time traffic forecasting and travel times estimation. See a detailed survey in chapter 5.

Soft-computing methods such as neural networks and support vector regression usually have various meta-parameters which should be properly set in order to achieve the best performance. The performance also strongly depends on the selected input variables. The reason is that many soft-computing methods can not work with missing inputs, or when some inputs can contain unimportant noise, which can deteriorate the quality of prediction. Hence automated multi-objective optimization methods are highly requested to simultaneously optimize relevant conflicting design objectives (such as parameters of soft computing models, selection of input data sensors etc.).

In the multi-objective optimization, it is possible to optimize two or more objective functions simultaneously. One of the difficulties of the multi-objective approach is how to compare two candidate solutions. This is because one solution can be good in one objective and worse in another and opposite. In order to deal with this problem, the Pareto dominance relation is often used. In contrast to a single objective optimization, for which the result is one single solution with the best value of the objective function, there are two goals for the multi-objective optimization. The first goal is to find solutions which are Pareto-optimal and the second goal is to find many different compromise solutions. In the past, it was shown that the optimization method called genetic algorithms are very successful in dealing with multi-objective optimization problems. It is mainly because genetic algorithms internally work with a set of candidate solutions. The quality of these candidate solutions is evaluated in each iteration of the algorithm and the new set of candidate solutions is generated based on the most perspective solutions from the last iteration. Finally, because huge volumes of data are processed, the involved soft computing methods as well as the meta-optimization methods working over these methods have to be carefully implemented in order to minimize the execution time.

1.1 Goals of the Thesis

On the basis of the previous brief survey of problems relevant for soft computing methods used in road traffic prediction, we have identified an open problem which is almost untouched neither in the literature nor practice:

An efficient approach is missing which will allow the soft computing methods to be automatically calibrated and utilized with the most suitable data samples in order to maximize the target quality measure.

The approach proposed to solve this hard problem, which is developed in this thesis, is based on multi-objective evolutionary algorithms. In order to demonstrate the effectiveness of the proposed approach, it will be evaluated on several case studies and compared with relevant methods. In the thesis, we define two main goals:

Goal 1: To propose a general framework for applying the multi-objective evolutionary optimization paradigm in the context of soft computing methods used in the area of traffic prediction and travel times estimation.

Goal 2: To evaluate the proposed framework using selected case studies. The subgoals are defined as follows:

Goal 2.1: To propose a new method for estimation of missing values in traffic density maps using a multi-objective genetic algorithms and compare this method with a conventional quadratic programming approach on real world data.

Many intersections and roads in modern cities are equipped with some kind of measurement device (traffic sensor). Unfortunately, it is very expensive to cover each road and each intersection by these devices. In this thesis, we propose a new method to estimate the values of traffic intensity on the roads which are not covered by traffic sensors. This method is based on a multi-objective genetic algorithms and is capable to provide better estimation than traditional quadratic programming approach for some situations. The main advantage of the new method is that the knowledge of a traffic expert can be incorporated into the process of estimation. Further, we propose a method which combines the quadratic programming approach and multi-objective genetic algorithm.

Goal 2.2: To propose a new multi-objective method for selection of input data (sensors) for support vector regression, in the task of short time traffic forecasting.

The machine learning methods such as support vector regression are quite sensitive to the proper selection of input variables. This problem is known as the feature selection. The good feature selection is even more important for the traffic prediction, because a huge amount of data can be unavailable. This can be caused by sensor malfunction, or by many other reasons. And using a sensor, which is very often broken as the input of machine learning method can cause that the traffic prediction system is unavailable. On the other hand, the selection of a very small subset of input variables (sensors) can lead to a very inaccurate prediction.

Proposed method is capable to find a proper subset of input variables for support vector regression (SVR). It is based on multi-objective genetic algorithm and especially useful in the scenarios with many missing data. The multi-objective genetic algorithm provides vari-

ous trade-off solutions between the SVRs requiring many input variables (which provides a very precise prediction) and less precise SVRs utilizing only several input variables. We also propose a method enabling to dynamically switch among these SVRs during the prediction process. The most precise SVRs are then used in the situation, when the complete data are available and the less precise SVRs are employed in the situations when many of input data streams are missing.

Goal 2.3: To evaluate the method for multi-objective selection of inputs for support vector regression in the task of travel times forecasting.

The objective is to utilize the method introduced in the previous paragraph for the prediction of travel times. In this case, we will utilize the data from traffic sensors and the data from license plate reading systems as potential inputs. The principle of dynamic switching among the SVRs remains the same. The goal is to provide more accurate prediction, which is available for large portion of time.

Goal 2.4: To further improve the traffic prediction by simultaneous multi-objective optimization of input variables and meta-parameters of support vector regression.

The previous approaches have dealt with optimizing only the set of input variables, but the meta-parameters of SVR remained unchanged. However, this is far from the optimum, because for each SVR the optimal settings of the meta-parameters is different. In order to deal with this problem, we will simultaneously optimize the input variables and SVR meta-parameters.

Goal 2.5: To provide an efficient parallel implementation of proposed methods.

The optimization methods based on genetic algorithms are usually quite computationally expensive. Modern computers are equipped by hardware, which is capable to split computational effort among many processor cores. However, this requires a parallel algorithm, which is able to work on many processor cores. The objective is to design, implement and evaluate parallel implementations of proposed methods.

1.2 Structure of the Thesis

This thesis is organized as follows. The second chapter surveys categories of traffic data, data collection methods and traffic sensors technologies. The description of basic traffic variables is also given.

The third chapter describes the machine learning algorithms. The difference between the supervised and unsupervised learning scenario is described. The chapter is mainly focusesed on two machine learning algorithms often used for traffic prediction neural networks and support vector regression.

The fourth chapter deals with the multi-objective optimization and multi-objective genetic algorithms. There are described the goals of multi-objective optimization and the techniques enabling to convert the multi-objective problem into a single objective problem. Then the description of major truly multi-objective genetic algorithms is given.

In the fifth chapter, we describe the types of traffic flow models, their structure and advantages and disadvantages. Then the summary of previously proposed machine learning techniques for traffic forecasting is provided.

Chapter 7 describes our new method for estimation of missing values in traffic density maps. The chapter also contains the comparison with quadratic programing approach.

The eighth chapter focuses on multi-objective optimization of input sensor selection for support vector regression. Here, the multi-objective approach is compared with the single objective optimization.

Chapter 9 describes the multi-objective selection of input sensors in the area of travel times forecasting. Our method is compared with the prediction based on the license plate reading. The goal is to improve the quality of prediction by simultaneous optimization of the input sensors and the SVR meta-parameters. The results for this new method are given in Chapter 10. The comparison with other methods is given in Chapter 11. Chapter 12 concludes this thesis.

Chapter 2

Traffic Data and Their Acquisition

The information about the traffic state is very important for drivers and traffic operators. The traffic data are also an essential input for intelligent transportation systems. The character and quality of traffic data are strongly dependent on the technology used for their acquisition. In this chapter, we survey types of traffic data and traffic variables. We also describe the most common technologies for traffic measurement and their advantages and disadvantages.

2.1 Types of Traffic Data

The most comprehensive type of traffic data is the trajectory data. This data contains the information about trajectories of all vehicles in the area. They are often observed using cameras installed in a helicopter or mounted on the top of buildings. Pictures provided by the cameras are processed by recognition software. Each vehicle is recognized and the information about its trajectory $x_\alpha(t)$ is provided. One camera covers only a small area of a few hundreds square meters at most. As the process of data collecting is very complex many errors can appear. For example, a car can be hidden behind another bigger car. The image recognition software can also make some errors. Because of these problems, the process of collecting reliable trajectory data is very expensive.

The collected trajectory data are often visualized in the form of a space-time diagram. An example of this diagram is shown in Figure 2.1. The horizontal axis represents time and the vertical axis shows the vehicle locations.

The space-time diagram gives important information such as the local speed of each vehicle, which is a gradient of the trajectory. The horizontal distance between two trajectories shows the time headway Δt_α which is the time distance between the front bumpers of two vehicles. The distance headway is the distance between two vehicles and is represented as a vertical distance of two lines in the time-space diagram. The lane changes are depicted as beginning and ending trajectories. Other macroscopic traffic variables such as traffic flow and traffic density can also be obtained [67].

The other type of traffic data is called the floating car data (FCD). In this case, selected vehicles are equipped with a system which is able to send information about the actual position of the vehicle. This is provided by GPS navigation, mobile phone or other kind of devices. Using obtained data, the trajectory of selected vehicles can be reconstructed. The selected vehicles can be additionally equipped with other sensors such as radars. These sensors provide the information about distance to the leading vehicle, or its speed. We

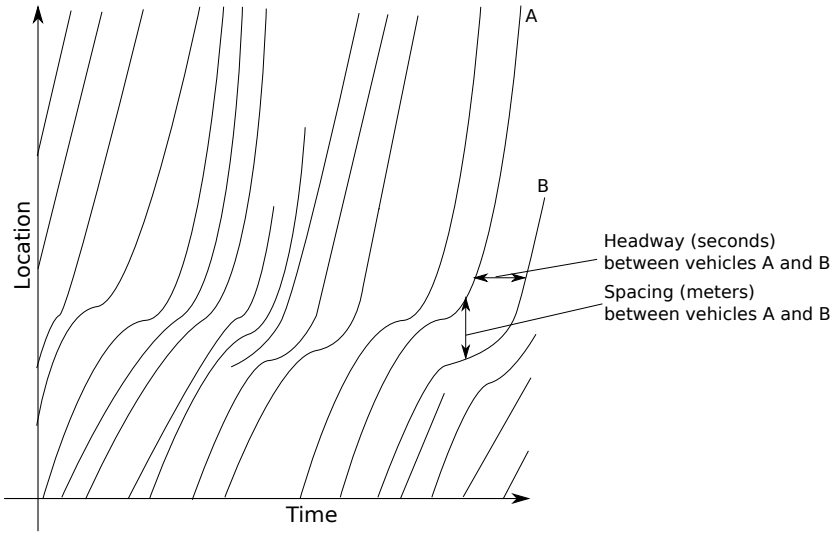

Figure 2.1: Time space diagram

call this kind of data as the extended floating car data (xFCD). The main difference between the FCD and trajectory data is that FCD data contains the information about only selected vehicles in the area. The information about the current line is not provided in FCD data. The reason is that GPS system is not precise enough to recognize the current line. Sometimes FCD are biased with respect to selected probe vehicles, for example, if the selected vehicles are public transport or taxies. On the other hand, FCD data can contain the information about the state of the vehicle and driver's behavior in the current situation [34, 81, 53].

The third type of data is called the cross-sectional data. This kind of data is provided by stationary sensors such as loop detectors, radars, cameras or infrared sensors placed on a certain location of the road. Provided information depends on the type of sensor. These sensors are transmitting either single vehicle data or data aggregated into time intervals [67].

2.2 Microscopic Traffic Analysis

The microscopic traffic data provides the most detailed information about the state of the traffic because they contain the information about each single vehicle. These data are often connected with some place of measurement on which the traffic sensor is located, or the place is interesting from the traffic analysis point of view. The basic information, which practically each measurement technology is able to detect is time t_α^0 in which the front of vehicle α passes the given place and time t_α^1, in which the rear end passes the given place.

Using this basic microscopic data, it is possible to calculate other microscopic quantities which are often called the secondary quantities. The length of vehicle l_α can be calculated as a time difference between the time when the front and tale of the vehicle passes the given place multiplied by the vehicle speed ($l_\alpha = v_\alpha(t_\alpha^1 - t_\alpha^0)$). Vehicle type is often detected by means of the length of vehicle. The time headway is a time distance between the front bumpers of successive vehicles ($\Delta t_\alpha = t_\alpha^1 - t_\alpha^0$). The distance headway is a distance between front bumpers of successive vehicles ($d_\alpha = v_{\alpha-1}\Delta t_\alpha$). Finally, the distance gap is the length between rear and front bumpers of neighboring vehicles ($s_\alpha = d_\alpha - l_{\alpha-1}$). [67]

2.3 Macroscopic Traffic Analysis

Collected data are often aggregated into time intervals having between 20 seconds to 5 minutes, such data are called the macroscopic data. The aggregation allows for the disc space and processing time reduction.

The traffic flow $Q(x,t)$ is the number of vehicles ΔN passing the current place x within a time interval Δt, i. e.

$$Q(x,t) = \frac{\Delta N}{\Delta t}. \tag{2.1}$$

The occupancy is a dimensionless variable representing the fraction of time interval during which a given place is occupied by vehicle. The occupancy is defined as

$$O(x,t) = \frac{1}{\Delta t} \sum_{\alpha=\alpha_0}^{\alpha_0+\Delta N-1} (t_\alpha^1 - t_\alpha^0), \tag{2.2}$$

where Δt is the length of the time interval, α_0 is the first vehicle which passed the current place in the given time interval, and ΔN is the number of vehicles which passed the given place during Δt, t^0 is the time when the front bumper of the vehicle passed the current place and t^1 is the time when the rare bumper of vehicle passed the given place. The arithmetic mean speed

$$V(x,t) = \frac{1}{\Delta N} \sum_{\alpha=\alpha_0}^{\alpha_0+\Delta N-1} v_\alpha \tag{2.3}$$

is a mean speed for ΔN vehicles passing the given place. v_α is the speed of vehicle α. The values of traffic flow, occupancy and arithmetic mean speed can be obtained by aggregating the variables directly measured by stationary traffic detectors. However, there also exist variables which are measured for the whole road segment. One of the most important ones is traffic density ρ:

$$\rho(x,t) = \frac{Q(x,t)}{V(x,t)} = \frac{flow}{speed} \tag{2.4}$$

which is a spatial average of the number of vehicles on a given road segment and speed V is a spatial average over the whole road segment [67].

2.4 Travel Times

Travel time τ_{12} is a time which a vehicle needs to pass a road section $[x_1, x_2]$. The value of travel time depends on the start and end position (x_1, x_2) and time t. Travel time is then defined as realized travel time $\tau_{12}(t_2)$:

$$\tau_{12}(t_2) = t_2 - t_1. \tag{2.5}$$

Another useful variable for traffic-flow optimization and congestion analyses is called total travel time τ_{tot}. It is a cumulative time

$$\tau_{tot}(x_1, x_2, t_1, t_2) = \int_{t_1}^{t_2} n_{12}(t)dt \tag{2.6}$$

spent by vehicles in the spatiotemporal region $[x_1, x_2] \times [t_1, t_2]$, where n_{12} is the number of vehicles on the section $[x_1, x_2]$ at time t [67].

2.5 Traffic sensors

Traffic detectors represent the traditional way of collecting traffic data [66, 70, 112]. There are two basic groups of traffic sensors: intrusive sensors and non-intrusive sensors. Intrusive sensors require the modification of the road surface and thus, in general, the installation of these sensors is not trivial. The non-intrusive sensors are installed somewhere near the road and don't require the modification of road surface. The sensors also differ in the utilized technology, quality of provided data and reliability. Depending on technology, these properties are affected by current weather or day time. In this section, we survey the most commonly used types of traffic sensors and discuss their advantages and disadvantages.

2.5.1 Inductive Loop Detectors

The inductive loops detectors were developed in the early 1960's. The inductive loop consists of one or more loops of insulated wire. This wire is installed in the shallow slot under the road. When a vehicle passes the place where the loop detector is located, the inductance of the loop decrease. This decrease of inductance is then detected by the detector electronic and can further be evaluated.

The data obtained by the inductive loops are further analyzed by modern pattern recognition algorithms which are able to provide the speed, the number of axes, direction and the size of vehicle. The main advantages of inductive loops are the precision and reliability, even in bad weather conditions [66, 70].

2.5.2 Piezoelectric Detectors

The piezoelectric detectors utilize the principle of the piezoelectric effect, i.e. generating voltage during deformation of a suitable crystal (e.g. quartz). The piezoelectric detector is usually placed under the surface of the road and measures the dynamic changes of the pressure produced by vehicles.

The main advantage of piezoelectric sensors is the ability to measure the weight of vehicles. The main disadvantage is the complicated installation requiring the modification of the road surface. Another disadvantage is the dependency of the output signals on temperature [82, 94].

2.5.3 Magnetometer Detectors

The magnetometer is a sensor of the size of a small can which is placed under the road surface. This sensor is able to detect the presence of ferrous metal of the vehicle. The magnetometer is used when only the information about the presence of the vehicle is needed [70].

2.5.4 Radar

Radars utilize the microwaves - the electromagnetic waves on the frequency higher than 300 MHz and lower than 300 GHz. These waves are sent out towards the vehicles. When

the wave reaches the vehicle, the signal is reflected back to the sensor. The reflected signal is then detected and further processed.

There are two basic types of radars. The continuous wave radar (CW) emit the continuous signal on the consistent frequency. This signal is reflected back by the moving object. The frequency of the reflected signal is modified by the Doppler effect according to the speed of the moving object. The second type of radar emits frequency modulated continuous wave (FMCW). These radars work with a signal, whose frequency is increasing during the time. The reflected signal is compared with the signal which is actually emitted. The difference between these two signals is then used to calculate the speed of moving objects.

The main advantage of the radar is the easy installation. It is possible to install it without any modification of the road surface. The radar is independent to weather and works very well during the day and night. Some types of modern radar detectors are even capable of measuring more road lines by one radar device. This can be a much cheaper alternative to the installation of loop detectors for each line. The disadvantage of the radar is its unreliability in the dense traffic as the measured intensity can be distorted in this situation [17, 61, 70].

2.5.5 Infrared Detectors (Laser)

The infrared detectors, also known as laser detectors (Light Amplification by Stimulated Emission of Radiation), are able to provide other non-intrusive technology to measure the traffic variables. The laser light has some specific properties which are very desirable in measuring of physical variables. The main property is rays coherency. It means that all rays are spreading into the one single direction. Another property is that the laser light is monochromatic.

The traffic sensors based on lasers emit the laser light towards the vehicles. This light is reflected back and detected by the optical system. The main advantage of laser sensors is that it is not necessary to modify the surface of the road when they are installed, the detectors work in day and night and provide very precise measurement of the vehicle speed. The main disadvantage is higher dependency on the weather conditions. In the case of strong rain, snow, or in the heavy fog, the precision of measurement can be deteriorated [68, 70].

2.5.6 Camera Detectors

The camera systems are widely used to monitor road traffic by human operators. Of all the sensor types the camera signal contains the most comprehensive information about the traffic situation in the given place. The data obtained by these camera systems can be processed by modern computer vision algorithms in order to extract the values of traffic variables.

The main advantage of camera detectors is that they can provide additional information about the traffic, which is impossible to detect by other technologies. It can be, for example, the license plate number of currently passing vehicle, the type of the vehicle and many others. On the other hand, the quality of provided information is strongly dependent on the weather conditions and the quality of the computer vision algorithms [19, 132, 94].

Chapter 3

Machine learning algorithms

Soft-computing methods are capable of dealing with uncertainty and problems showing high complexity. These kinds of problems often appear in the area of image processing, signal processing, multi-objective optimization and many others. In this thesis, we mainly utilize two types of soft-computing methods. The first type is machine learning based on neural networks and support vector regression and the second type is a genetic algorithm. The principles of machine learning, neural networks and support vector regression are described in this chapter. Genetic algorithms will be described in the next chapter where a special attention will be given to the multi-objective genetic algorithms.

3.1 Supervised learning and prediction

Machine learning algorithms are capable of adapting their behavior according to the input data. There exist three basic types of machine learning algorithms. In the case of supervised learning, the machine learning model is trained using these samples. Then the model can be used to provide answers for new samples. The second type of machine learning algorithms uses unsupervised learning technique. The input data doesn't contain the correct answers and the goal of the learning process is to find similarities in data and select the correct categories for samples. Finally, in reinforcement learning, the algorithm gets to know when the answer is wrong, but does not get the correct answer. In this thesis, we will focus mainly on the supervised learning techniques.

In the supervised learning scenario, we have given a set of pairs $D = \{(\vec{x}^{(n)}, \vec{y}^{(n)}), n = 1 \dots N\}$. The goal of the learning is to find mathematical dependency between the input $\vec{x}$ and the output $\vec{y}$ in such manner that if a new value $\vec{x^*}$ is given, the calculated value $\vec{y^*}$ is shows as small error as possible. The new tuple $(\vec{x^*}, \vec{y^*})$ may not be in the set D, but has to be generated by the same process as the members of D. In the case the value of $\vec{y}$ belongs to one of a few discrete values, the process is called classification. In the case the value of $\vec{y}$ is continuous, the process is called regression.

The supervised learning process has three phases. These phases are depicted in Figure 3.1. At first, it is necessary to split the set D into two parts. In the phase one, the first part D_{train} is used to train the machine learning model. Then it is necessary to validate the quality of the trained model using set D_{test} (phase 2). If the model provides results of sufficient quality then it is ready for the use (phase 3) [91, 128].

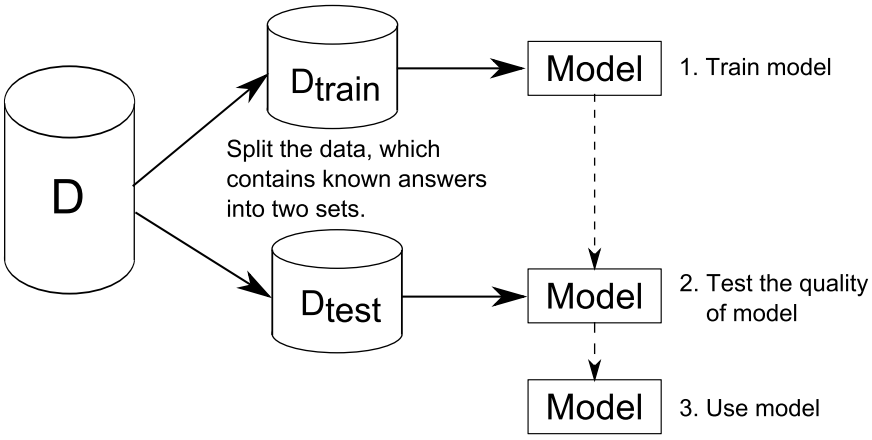

Figure 3.1: The process of supervised learning.

Many quality measures were proposed for evaluation of prediction results. One of them is the maen absolute error (MAE)

$$MAE : \frac{\sum_{i=1}^{N} |y_i - y_i'|}{N}, \tag{3.1}$$

where y_i represents the correct value, y_i' represents the predicted value and N is number of samples. Another widely used measure is the root mean squared error (RMSE)

$$RMSE : \sqrt{\frac{\sum_{i=1}^{N} (y_i - y_i')^2}{N}}. \tag{3.2}$$

which calculates the square of differences between real and predicted values.

The two above measures provide results in the same unit scale as the predicted variable. However, sometimes it is useful to compare the results of the prediction with a naive predictor, which always returns the mean of the predicted variable $\bar{y}$. Hence, the relative absolute error (RAE)

$$RAE : \frac{\sum_{i=1}^{N} |y_i - y_i'|}{\sum_{i=1}^{N} |y_i - \overline{y}|}, \tag{3.3}$$

relative squared error (RSE)

$$RSE : \frac{\sum_{i=1}^{N} (y_i - y_i')^2}{\sum_{i=1}^{N} (y_i - \overline{y})^2} \tag{3.4}$$

and root relative squared error (RRSE)

$$RRSE : \sqrt{\frac{\sum_{i=1}^{N} (y_i - y_i')^2}{\sum_{i=1}^{N} (y_i - \overline{y})^2}} \tag{3.5}$$

were proposed. If the value outputed by these metrics is higher than 1, the tested predictor is worse than the naive predictor. Otherwise, the predictor is better than the naive one [46].

3.2 Data preprocessing

The real world data are very rarely in the form which is suitable for machine learning algorithms. They are incomplete, noisy or inconsistent. Preprocessing methods are employed to transform the data into a suitable form. The data preprocessing methods can be divided into several groups:

- Data cleaning, which tries to fill the missing values in data, smooth out noise and identify outliers.
- Data integration merges data comming from different sources.
- Data transformation transforms the data into the form suitable form for machine learning algorithms.
- Data reduction enables to obtain reduced representation of data.

3.2.1 Data cleaning

One of the most important tasks in the data cleaning process is how to deal with missing values. This problem is especially important in the case of traffic, because the data obtained from sensors often contains missing values. Probably the easiest way how to deal with sample, which contains missing values is to ignore it. This approach can be used in the situation in which the portion of missing values is not huge. Other approaches try to somehow fill the missing values. For example, the user can fill the missing values manually according to his expert knowledge. It is a very time consuming process and it is thus used quite rarely. Another method fills the missing values by constant. However, it is quite hard to choose a proper constant. Most sophisticated methods dealing with missing values are based on regression. These methods try to find mathematical dependencies between available values and predict the missing values.

Another important task in the data cleaning process is noise removal. The noise is seen as a random error in the measured variable. For example, in the case of averaging, selected values are replaced by the mean of the bin (a subset of data) which they belong to.

Outliers are data samples whose values are completely different to the remaining values. The outliers can significantly worsen the performance of the machine learning algorithms. So, it is desirable to find the outliers and remove them from the data. One of the approaches to find the outliers is based on clustering. In this method, the data samples are organized into groups according to similarity. The data samples outside these groups are considered as outliers [128, 46].

3.2.2 Data integration

The real world data are often located in completely different databases and have various structures. For example, traffic data for one area can be obtained by various types of traffic sensors and can be located in many different information systems. The process of merging these data is called the data integration.

During this process, various problems can appear such as schema integration problems and object matching problems. In the schema integration problem, the same data can be organized in different data structures, have different aggregation level, or the variables can have a slightly different meaning. The object matching problem can appear when it is hard

to recognize the same entity. For example, whether the data were measured for the same crossroad.

Data obtained from different sources often contains redundancy. For example, the same traffic variable can be measured by two different sensors located at the same place. An important tool enabling to detect the redundancy is the correlation analysis. It enables us to measure how strongly one variable implies the other, by computing the correlation coefficient such as Pearson's coefficient

$$r_{A,B} = \frac{\sum_{i=1}^{N}(a_i - \overline{A})(b_i - \overline{B})}{N\sigma_A\sigma_B}, \tag{3.6}$$

where N is the number of samples, a_i and b_i are values of variables A and B. The σ_a and σ_b are the standard deviations of A and B. $r_{A,B}$ belongs to interval $[-1, 1]$. If the value of Pearson's coefficient is close to 1 then the variables A and B are positively correlated. The value close to zero indicates that variables A and B are not correlated and if the value of $r_{A,B}$ is close to -1, the variable A and B are negatively correlated.

If one variable is represented in the data multiple times, it is necessary to have a mechanism capable of resolving potential conflicts. For example, in traffic data we can have two or more different sensors measuring the same variable on a given place. When these sensors are providing different values, the mean can be taken as the resulting value. If it is known that one sensor is more reliable than the other, the more reliable one is taken. It is also possible to allow the human operator to resolve the conflict [46].

3.2.3 Data transformation

The machine learning and soft-computing algorithms usualy require the data in a specific format. Neural networks and SVR, which are described in the following chapter, work well when the input data are normalized. The normalization is a process transforming a given variable into a predefined range. The min-max normalization is very often utilized. It performs a linear transformation on the input data defined by the following equation

$$x' = \frac{x - min}{max - min}(MAX - MIN) + MIN, \tag{3.7}$$

where x' is the normalised value of x. The min and max represents the minimum and maximum of the former variable. The MIN and MAX represents the minimum and maximum after the normalisation.

Another normalisation approach is called the zero mean normalization. This approach has some very desirable properties, for example, normalised values are centralised around the zero and the standard deviation equals to one. The normalisation is defined by

$$x' = \frac{x - \bar{x}}{\sigma}, \tag{3.8}$$

where x' represents the normalised value of x, the $\bar{x}$ represents the mean and σ is the standard deviation [128, 46].

3.2.4 Data reduction

In the field of road traffic analysis, we process large collections of data. It is often necessary to reduce the volume of this data, or to select a reasonable part of it. There are two general ways for doing so.

The first is to reduce the number of data points. For example, we can take data from 14 days instead of one month for the analyses. Or, we can aggregate measured values into 5 minute intervals, instead of one minute intervals. This data reduction can also significantly reduce the time of computation for machine learning algorithms.

The second way to reduce the amount of data is to reduce the number of input variables for the model. This can be done by selection of proper model inputs, which is called feature selection. Or, it can be done by deriving new features from existing ones. This process is called feature derivation. The feature derivation is often based on applying transforms to the original dataset that change the coordinate system. The reason for such transformation is that some dimensions in the new coordinate system can be redundant or have only a limited impact on the learning process.

Linear Discriminant Analysis (LDA) is the example of a supervised method for feature derivation [37]. The main idea of LDA is to transform the data into the coordinate system, where the within class scatter is small and the between class scatter is large. The within class scatter is defined as the covariance of each class with itself:

$$\sum_{\vec{x} \in c} (\vec{x} - \vec{\mu}_c)(\vec{x} - \vec{\mu}_c), \tag{3.9}$$

where $\vec{x}$ is a data point from class c and $\vec{\mu}$ is a vector containing the mean values for each variable. By weighted suming the values for all classes, we get the within-class scatter of the whole dataset:

$$S_w = \sum_{c \in classes} p_c \sum_{\vec{x} \in c} (\vec{x} - \vec{\mu}_c)(\vec{x} - \vec{\mu}_c). \tag{3.10}$$

where p_c is the probability of class c and is equal to the number of data points for class c divided by the number of all data points. The between class scatter is calculated as

$$S_B = \sum_{c \in classes} (\vec{\mu}_c - \vec{\mu})(\vec{\mu}_c - \vec{\mu}), \tag{3.11}$$

where $\vec{\mu}_c$ represents a vector containing means for all variables within the whole dataset.

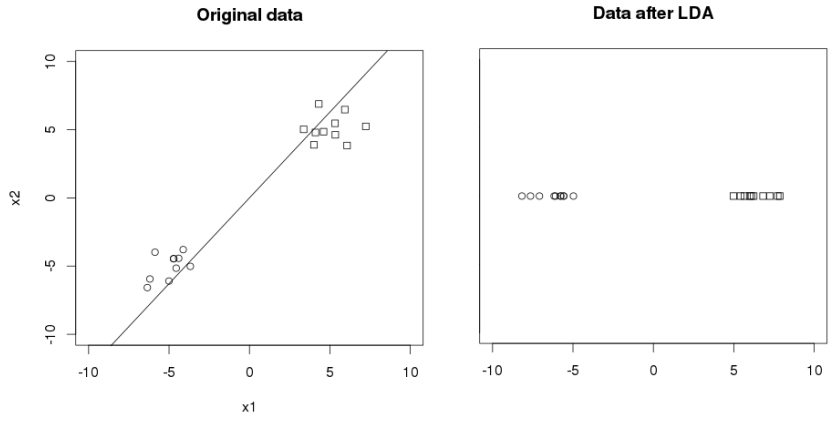

Figure 3.2: Example of dimensionality reduction using Linear Discriminant Analysis: The original data in two dimensional space (left). The data reduced into one dimensional space (right).

An example of dimensionality reduction using LDA is depicted in Figure 3.2. It can be seen that it is possible to separate the data in this one dimensional space.

Another approach, which is often used to reduce the data dimensionality is Principal Component Analysis (PCA) [62]. This method is unsupervised, which means, that we don't need to know the class labels for data points. The principal component is a direction in data with the largest variation. The algorithm finds the direction with the largest variation in data and places the first axis in that direction. Then it looks for the remaining variation and defines the next axis, which is orthogonal to the first axis. The algorithm continues until all remaining axis are defined. After the transformation is performed each new variable is unrelated with other variables. It means that the covariance matrix is diagonal.

An example of dimensionality reduction performed by PCA is depicted in Figure 3.3. The original data are depicted on the left side. The data consists of two easily separable groups of data points. The figure in the middle shows the data transformed into one dimensional space. It is important to note that the method is unsupervised and it knows nothing about the groups. On the right, there are depicted the data from the middle picture after the backward transformation was performed.

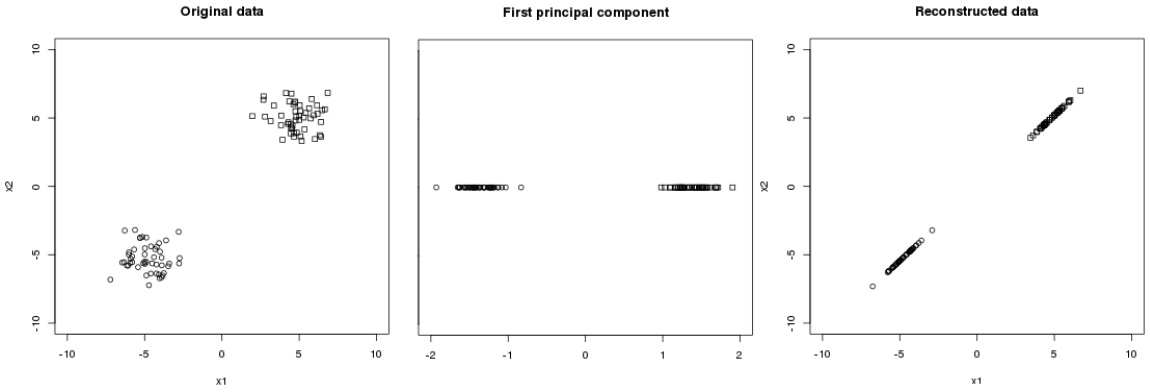

Figure 3.3: Example of dimensionality reduction by Principal Component Analysis: the original data (left), data after the reduction in one dimensional space using PCA (center), data reconstructed back to the two dimensional space (right).

Another approach for the dimensionality reduction is called discrete wavelet transformation, which is popular in signal processing. This method is able to transform the data vector into the vector of wavelet coefficients. The dimensionality reduction is based on the fact that some of these coefficients are more important than others. In the past, many other methods for the dimensionality reduction were proposed [12], for example, independent component analysis [121, 57], locally linear embedding [110], or Isomap [127].

3.3 Neural networks

The artificial neural networks are inspired by the behaviour of neurons in animals and human brain. These neurons are largely simplified against the neurons in brain. Also topology of the artificial neural network is simplified against the real brain. Various kinds of artificial neural networks were proposed. They primarly differ in modeling of neurons, topology of network and learning algorithms [92, 91].

In the following text we will focus on neural networks with the backpropagation learning

mechanism. In these networks, each neuron has several inputs and one output. The output of the neuron is calculated using two functions. The first one is called the basis function ξ and is defined as the weighted sum of inputs:

$$\xi = \sum_{i=0}^{N} (w_i x_i), \tag{3.12}$$

where w_i is weight for the input x_i, N is the number of neuron inputs, and w_0 is considered as a treshold value.

The output ξ of the basis function serves as the input for the activation function usually defined as

$$O = \frac{1}{1 + e^{-\xi}}. \tag{3.13}$$

The output of the activation function is the output of the neuron. A model of the whole neuron is depicted in Figure 3.4.

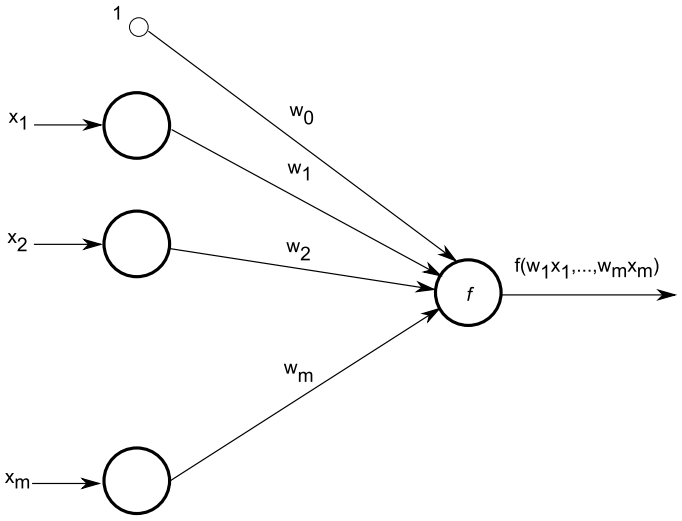

Figure 3.4: The model of artificial neuron.

The neural network is an oriented graph in which the nodes represent neurons and the oriented edges represent interconnections between the neurons. There are three types of layers in neural networks: the input layer, one or more hidden layers, and the output layer. The global outputs of the whole network are connected to neurons of the output layer. An example of topology of a feed forward neural network is depicted in Figure 3.5.

The backpropagation algorithm is one of learning algorithms for feed forward neural networks. Its formal notation can be found in [92, 91]. For purposes of this thesis, we will only sketch its basic idea. The learning starts with a random initialization of the weights and basis. Then, for each tuple in training set $X_i \in D_{train}$, the following process is performed. The input values are connected to the neurons of the input layer. The number of these neurons should be the same as the number of inputs. The input neurons simply return the current input values into their outputs. The outputs of input neurons serve as the input for neurons in the first hidden layer. Then neurons in the hidden layers and output layer use the basis and activation functions to calculate their outputs.

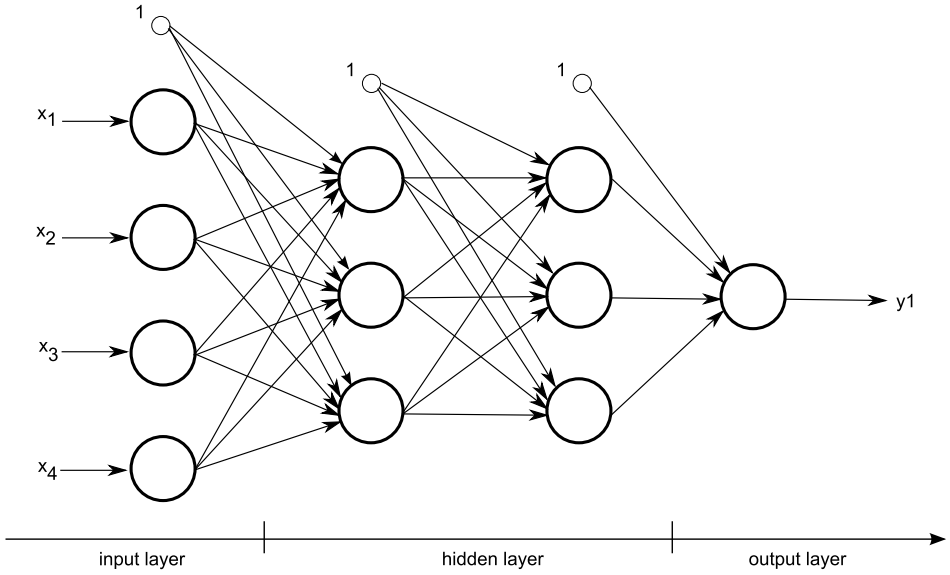

Figure 3.5: The topology of the feed forward neural network.

The output values are compared with desired output values. The difference between the desired values and values provided by the output neuron j is called error E_j. The error is then propagated backward from the output neurons to the neurons in the first hidden layer of the network. After the error is calculated for each hidden neuron, weights are updated. An important factor in the weights updating is learning rate l. This rate defines the speed of weights update. If the learning rate is small, the weights are updated only slightly during the learning process. On the other hand, if the learning rate is high, the values of weights are updated more agresively. The typical value of learning rate is lower than 1 and its proper setting is usually very dependent on the task. After the weight updating is repeated for all items in the training set, the termination condition is evaluated. If the termination condition is satisfied, the learning process ends. Otherwise, the process is repeated for all items of the training set [92, 91].

3.4 Support vector machine and support vector regression

The support vector machine (SVM) is a very popular soft computing method for solving classification tasks [11]. It was successfully used in the area of computer vision [55, 109], handwriting recognition [147], bioinformatics [107], economy [56] and many others. The basic variant of SVM is capable of solving only linearly separable problems. The algorithm becomes more powerful when the so-called kernels are introduced. Kernels are special functions capable of transforming input data into a more dimensional space. This transformation allows SVM to successfully solve non-linear problems, because in these more dimensional spaces data can often be separated.

However, the tasks discussed in this thesis are mostly regression tasks and SVM algorithm is not able to solve them. Fortunately there exists a modification of SVM - support vector regression (SVR) which is able to solve regression tasks. SVR appears to be very useful in forecasting of time-series such [126, 14, 141].

Both SVM and SVR have various meta-parameters which have a significant impact to classification and regression quality. It is often very difficult for human to set these

parameters properly. In the past, various methods to solve this problem were proposed [95, 141]. We will discuss linear SVM, SVM with kernels, SVR and SVM/SVR parameter optimization in the following sections.

3.4.1 Linear support vector machine

Before we will describe the basic principles of SVM, it is necessary to define the linearly separable problem. Using the same notation as in Section 3.1, we will expect the classification problem in the form $\{(\vec{x}_i, y_i), i = 1 \dots N\}$. In our case, $\vec{x_i}$ is a vector of real numbers and y_i represents the correct classification of the item. Assume that there are two classes marked by values -1 and 1 ($y_i \in \{-1, 1\}$). The problem in this form can be considered linearly separable, if hyperplane $\vec{w}.\vec{x} + b = 0$ exists, which separates the points labeled -1 from points labeled 1. Here, $\vec{w}$ is a normal to hyperplane and $b/\|\vec{w}\|$ is the perpendicular distance from the hyperplane to the origin. The notation $\|\vec{w}\|$ represents the Euclidian norm of $\vec{w}$.

In the case of linearly separable problems, the SVM looks for separating hyperplane with the largest margin. To find this hyperplane, it is necessary to solve the following optimization problem

$$\begin{aligned} &\text{minimize:} && \tfrac{1}{2}||\vec{w}||^2 \\ &\text{subject to:} && \begin{cases} \vec{x}_i.\vec{w} + b \geq +1 & \text{for} \quad y_i = +1 \\ \vec{x}_i.\vec{w} + b \leq -1 & \text{for} \quad y_i = -1 \end{cases} \end{aligned} \tag{3.14}$$

The constrains here guarantee that the hyperplane will separate the classes and the optimized function $\frac{1}{2}||\vec{w}||^2$ will provide vector $\vec{w}$, for which the margin between the samples of two classes is maximal. In the training set, there exist special points, for which the dot product $\vec{x}_i.\vec{w}$ is equal to 1 or -1. These points are called support vectors.

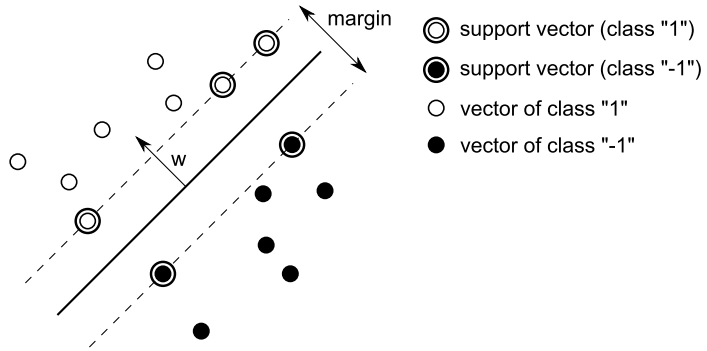

Figure 3.6: Example of linearly separable problem.

However, non-trivial real world classification problems are not linearly separable and it is useful to tolerate some small amount of misclassified training samples. In order to solve linearly non-separable problems, it is necessary to reformulate the constrains by introducing the concept of slack variables (ξ_i). These slack variables allow some amount of misclassified samples in exchange for an increase of the value of optimized function. The SVM is then re-defined to:

$$\begin{aligned} \text{minimize:} \quad & \tfrac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{l}(\xi_i) \\ \text{subject to:} \quad & \begin{cases} \vec{x}_i.\vec{w} + b \geq +1 & \text{for} \quad y_i = +1 - \xi_i \\ \vec{x}_i.\vec{w} + b \leq -1 & \text{for} \quad y_i = -1 + \xi_i \\ \xi_i & \geq \quad 0 \end{cases} \end{aligned} \tag{3.15}$$

Coefficient C is set by the user and defines the trade-of between the possible amount of misclassified samples and the size of margins. The principle of slack variables is depicted Figure 3.7.

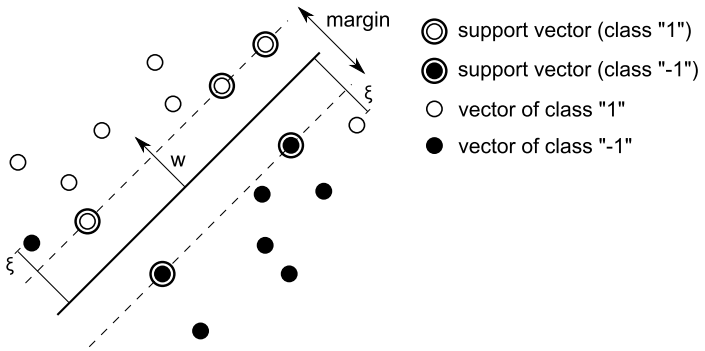

Figure 3.7: Example of SVM applied to a problem, which is not linearly separable.

3.4.2 Non-linear support vector machine

The linear SVM can be very useful, but many classification tasks don't have a linear character. To deal with these problems, the transformation of independent variables into more dimensional space is commonly used. To do this we need a proper function in the following form $\Phi : \mathcal{X} \to \mathcal{F}$, where $\mathcal{F}$ has more dimensions than $\mathcal{X}$. In the case of SVM, these mapping functions are called kernels. These kernel functions replace the dot product in previously described SVM. In the past, several kernels were proposed, including polynomial kernel (eq. 3.16), radial kernel (eq. 3.17) and sigmoid kernel (eq. 3.18). Parameters of these kernels will be discusssed in Chapter 10.

$$K(\vec{x}, \vec{y}) = (\vec{x}.\vec{y} + 1)^p \tag{3.16}$$

$$K(\vec{x}, \vec{y}) = e^{\frac{-||\vec{x}-\vec{y}||^2}{2\sigma^2}} \tag{3.17}$$

$$K(\vec{x}, \vec{y}) = tanh(\kappa\vec{x}.\vec{y} - \delta) \tag{3.18}$$

3.4.3 Support vector regression

The modification of SVM for regression tasks is called support vector regression (SVR). The training of SVR can be considered as an optimization problem [118], [5]. The goal of this optimization problem is to find a function $f(x)$ such that it has at most ε deviation from the correct output y_i for the given training vector $\vec{x_i}$. At the same time we want this function as flat as possible. In the case of SVR, the flatness mean the minimization of vector $\vec{w}$. More precisely, it means the minimization of $\frac{1}{2}||\vec{w}||^2$.

For many tasks, however, this problem can be infeasible. It means there is no such $\vec{w}$, for which the deviation from the correct y_i is at most ε. Because of it, the so called „soft margin loss function“ was introduced. New slack variables ξ and ξ^* are used to convert the infeasible problem into the feasible one. The precise formulation of SVR is given by equation:

$$\begin{aligned} \text{minimize:} \quad & \tfrac{1}{2}||\vec{w}||^2 + C\sum_{i=1}^{l}(\xi_i + \xi_i^*) \\ \text{subject to:} \quad & \begin{cases} y_i - \vec{w}.\vec{x}_i - b & \leq & \varepsilon + \xi_i \\ \vec{w}.\vec{x}_i + b - y_i & \leq & \varepsilon + \xi_i^* \\ \xi_i, \xi_i^* & \geq & 0 \end{cases} \end{aligned} \tag{3.19}$$

where $\vec{w}.\vec{x}_i$ is a dot product of vectors $\vec{w}$ and $\vec{x}_i$. The regularization meta-parameter $C > 0$ controls the trade-of between the flatness and the amount of deviation larger than ε. The situation is depicted in Figure 3.8, where the samples from the training set are marked as small crosses and the regression function is shown as the bold line. The permitted deviation is depicted by two thin lines. However, some points from the training set do not lie in the permitted area. The sizes of slack variables ξ and ξ^* for these points are also shown Figure 3.8. This basic version of SVR can successfully solve only linear tasks. Fortunately the same kernels that we have introduced SVM can be used to modify SVR in order to deal with non-linear regression tasks.

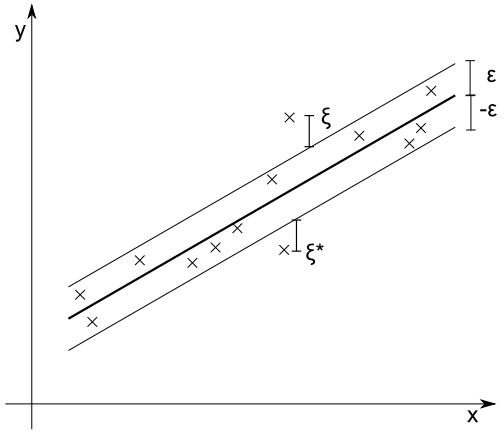

Figure 3.8: Support vector regression.

3.4.4 Kernel function and parameter optimization

As mentioned in the previous section, SVM and SVR have various meta-parameters, such as the kernel function, regularisation parameter and other parameters related to currently used kernel. It is necessary to set up these meta-parameters properly in order to obtain the best classification or regression quality [32]. In the past, several conventional optimization techniques and guidelines were proposed to solve this problem [86]. Some of these approaches are based on the principles of evolutionary algorithms [54, 140, 141].

For example, the HGA-SVR is a method for kernel function selection and parameter optimization in SVR. This method is based on genetic algorithm which simultaneously optimizes the type of kernel and meta-parameters of SVR. In case of HGA-SVR, each chromosome consists of the integer part and real valued part. The integer part has one

value, which defines the kernel type: linear kernel (0), polynomial kernel (1) or RBF kernel (2). The real valued part of the chromosome specifies the values of SVR meta-parameters. This method was originally designed to predict maximal daily electric load where provided better generalization capability and a lower prediction error than other approaches based on neural networks or traditional SVR [141].

Chapter 4

Multi-objective optimization using evolutionary algorithms

4.1 Multiobjective optimization problem

In the area of single objective optimization, the quality of a candidate solution is defined using one objective function f. The goal of the single objective optimization is to find a solution with the minimal or maximal value of a given function f. However, many real world optimization problems can't be described using only one objective function and it is necessary to use more objective functions $f_1, \ldots, f_m$, each of which has to be minimized or maximized. In this case, we speak about a multi-objective optimization. In many cases, objective functions are conflicting, i.e. improving one objective means worsening the other one. More formally, the multiobjective optimization problem (MOOP) is defined as follows [21]:

$$\begin{aligned} \text{minimize: } & f_m(\vec{x}), && m = 1, 2, ..., M \\ \text{subject to: } & x_i^{(L)} \leq x_i \leq x_i^{(U)}, && m = 1, 2, ..., M \\ & g_j(\vec{x}) \geq 0, && j = 1, 2, ..., J \\ & h_k(\vec{x}) = 0, && k = 1, 2, ..., K. \end{aligned} \tag{4.1}$$

A candidate solution $\vec{x}$ is a vector of n decision variables $\vec{x} = (x_1, \ldots, x_n)$. Functions $f_1, \ldots, f_m$ represent objective functions. Each component x_i of vector $\vec{x}$ must be within the range $x_i^{(L)}$ and $x_i^{(U)}$. We call values $x_i^{(L)}$ and $x_i^{(U)}$ as variable bounds. The functions g_i and h_i are called constraint functions, where g_j define inequality constraints and h_k define equality constraints. Solution $\vec{x}$ is feasible if it satisfies all constraints and is within defined variable bounds. Otherwise, $\vec{x}$ is an infeasible solution [21].

As an example of the multiobjective optimization problem, we can take a simple problem of two decision variables x_1 and x_2. It is requested that $x_1 \in [0.1, 1]$ and $x_2 \in [0, 5]$. It means, that $x_1^{(U)} = 1$, $x_1^{(L)} = 0.1$, $x_2^{(U)} = 5$ and $x_2^{(L)} = 0$. Two objective functions f_1 and f_2 have to be minimized:

$$f_1(\vec{x}) = x_1 \tag{4.2}$$

$$f_2(\vec{x}) = \frac{(1 + x_2)}{x_1}. \tag{4.3}$$

All feasible solutions have to fulfill constraints defined by functions g_1 and g_2:

$$x_2 + 9x_1 \geq 6 \tag{4.4}$$

$$-x_2 + 9x_1 \geq 1. \tag{4.5}$$

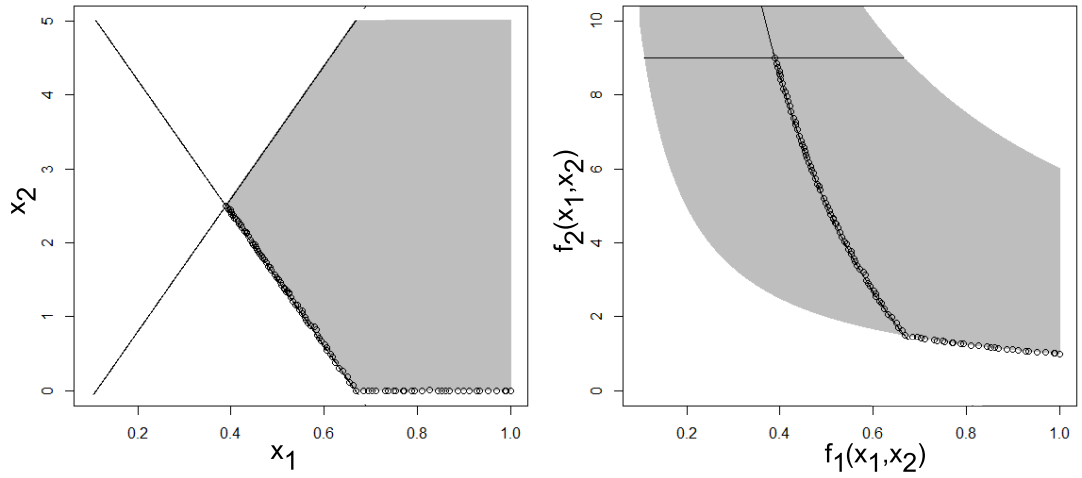

Figure 4.1: Parameter space (left) and objective space (right) [21].

Figure 4.1 shows the parameter space (x_1, x_2) and the objective space (f_1, f_2) for the considered problem. Constraint functions are represented by black lines and the area of feasible solutions is grey. Solutions obtained for this problem are marked as circles in both spaces. The result is no longer one solution, but a set of solutions.

4.2 Pareto dominance relation

One of the most important difficulties in the multiobjective optimization is how to compare the quality of candidate solutions. In the single objective optimization domain, the situation is quite straightforward. The solution $\vec{a}$ is better than solution $\vec{b}$ if the value of $f(\vec{a})$ is better than $f(\vec{b})$. We will denote the situation in which the value of function f for solution $\vec{a}$ is better than for $\vec{b}$ as $f(\vec{a}) \lhd f(\vec{b})$. In the case of minimization of function f, the statement $f(\vec{a}) \lhd f(\vec{b})$ would mean that $f(\vec{a}) < f(\vec{b})$. In the case of maximization, the same statement would mean that $f(\vec{a}) > f(\vec{b})$. Moreover, we will denote the situation in which the value of function f for solution $\vec{a}$ is not worse than for $\vec{b}$ as $f(\vec{a}) \not\rhd f(\vec{b})$. This means that $f(\vec{a}) \leq f(\vec{b})$ in case of minimization and $f(\vec{a}) \geq f(\vec{b})$ in case of maximization.

However, in the multiobjective optimization the situation is much more complicated. The main problem is that we have a vector with two or more values of objective functions for each candidate solution. Because of it, the Pareto dominance relation was established [21].

Definition 1 *A solution $\vec{a}$ is said to dominate the other solution $\vec{b}$ ($\vec{a} \preceq \vec{b}$), if both following conditions are met:*

1. *The solution $\vec{a}$ is not worse than $\vec{b}$ in all objectives ($f_i(\vec{a}) \not\triangleright f_i(\vec{b}), \forall i = 1, \ldots, M$).*
2. *The solution $\vec{a}$ is better than $\vec{b}$ in at least one objective ($\exists i \in \{1, \ldots, M\} : f_i(\vec{a}) \triangleleft f_i(\vec{b})$).*

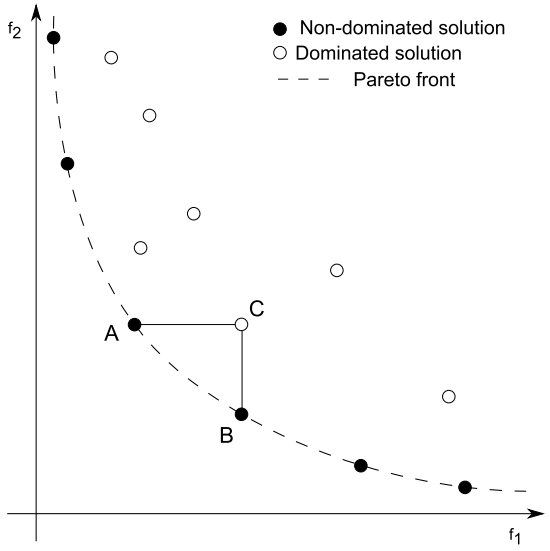

Figure 4.2: Examples of dominated and non-dominated solutions.

Figure 4.2 shows an example of Pareto dominated and non-dominated solutions. The depicted scenario expects minimization of two objective functions f_1 and f_2. We can say that solution A dominates C. The first condition is satisfied because $f_1(A) < f_1(C)$ and $f_1(A) = f_2(C)$. The second condition is satisfied, because $f_1(A) < f_1(C)$.

The Pareto dominance relation is transitive and is irreflexive. So, the Pareto dominance relation is a strict partial order. Figure 4.2 also shows a set of solutions as empty and filled circles. The most interesting subset of marked solutions consists of solutions which are not dominated by any other solution in the set (filled circles). According to the definition 2 we call these solutions non-dominated solutions.

Definition 2 *Among all solutions in set P, the non-dominated subset of solutions P' contains those solutions that are not dominated by any member of the set P.*

If P consists of all feasible solutions of the given problem, we call the non-dominated solutions as globally non-dominated solutions. The Pareto front is a curve which connects globally Pareto optimal solutions.

In the multi-objective optimization, we have two goals for optimization algorithms. The first is to find solutions lying on the Pareto front and the second is to obtain solutions widely spread along the whole Pareto front. This is a big difference against the single objective optimization where we have only one goal, which is to find a single best solution [21].

4.3 Transformation of multi-objective problems into single objective problems

In the past, many algorithms for single objective optimization were proposed. Thus, it is natural to try to convert a multi-objective problem to the single objective problem. In this section, two well-known conversion approaches are described. However, both require additional algorithm settings. And to obtain different solutions along the Pareto front, it is necessary to run the single objective optimization algorithm multiple times with many different settings.

4.3.1 Weighted Sum Approach

The weighted sum approach scalarizes the vector with values of objective functions to a single value using equation 4.6. Each objective function has assigned a weight. Then the value of each objective function is multiplied by the given weight. To obtain the scalarized value the weighted values must be summed.

$$F(x) = \sum_{m=1}^{M} (w_m f_m(\vec{x})) \tag{4.6}$$

The biggest difficulty of the weighted sum approach is how to set the weight vector $\vec{w}$ properly. In general, values of the weight vector should be higher for objectives of higher importance and opposite.

The concrete values are usually set by the author of the algorithm according to his problem knowledge and experience. Fortunately few rules about the values of weights were discovered. These rules mainly deal with a problem of Pareto optimality of obtained solutions. The rule 4.1 defines how to set the values of weights to obtain the Pareto optimal solution. Another rule 4.2 tells us when the Pareto optimal solution can be discovered using the weighted sum approach [21].

Rule 4.1 A globaly optimal solution to the problem represented by equation (4.6) is Pareto-optimal if the weight is positive for all objectives.

Rule 4.2 If $\vec{x}$ is a Pareto-optimal solution of a convex multi-objective optimization problem, then there exists a non-zero positive weight vector $\vec{w}$ such that $\vec{x}$ is a solution to the problem given by equation 4.6.

The big disadvantage of the weighted sum method is that it can not find solutions from the non-convex parts of Pareto front. The example of such situation is shown in Figure 4.3. Here, the non-convex part of the Pareto-front is marked by a dashed line, and the example of one solution, which it is impossible to find by the weighted sum method is marked by a filled circle.

To demonstrate basic principles of the method, a simple multi-objective problem will be discussed. The problem is defined by two objective functions f_1 and f_2.

$$f_1 = x^2 \tag{4.7}$$

$$f_2 = (x-2)^2 \tag{4.8}$$

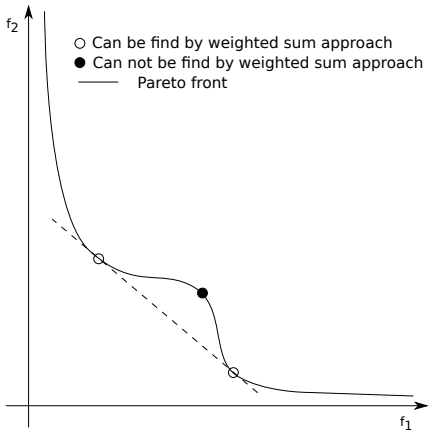

Figure 4.3: Examples of non-convex problem. Objective space is depicted.

The parameter vector has the size equal to one. The given problem has no constraints. The fitness functions are depicted on the top of Figure 4.4. To demonstrate the weighted sum approach we chose three weight vectors: $\vec{w}_a = (1, 0.5)$ gives more importance to the first objective, $\vec{w}_b = (1, 1)$ considers both objectives equally important, and $\vec{w}_c = (0.5, 1)$ considers the second objective as more important. The values obtained after the application of weighted sum are depicted on the left side of Figure 4.4.

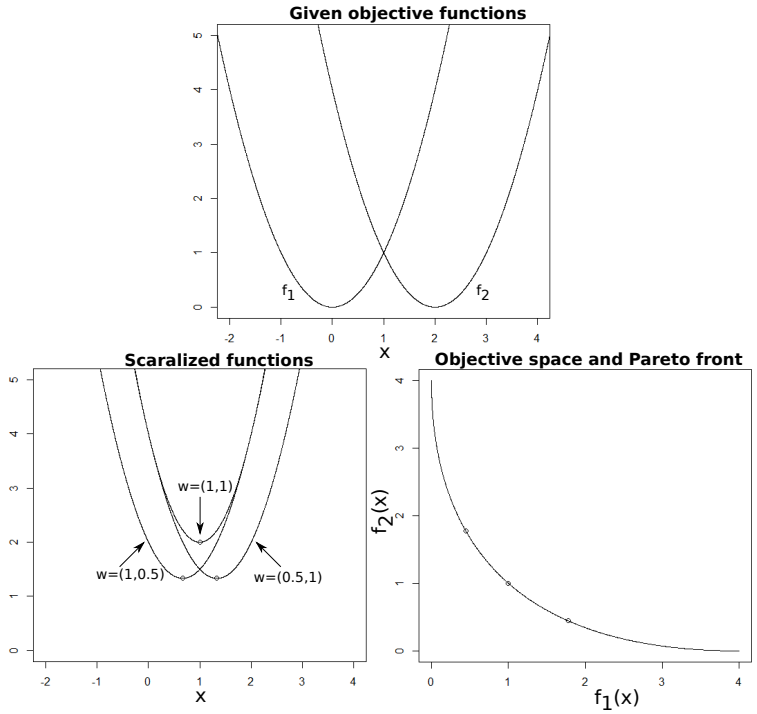

Figure 4.4: Solving example problem using Weighted Sum Approach.

The minimum for each weighted function is marked by a circle. The optimal solutions for each weighted vector are depicted on the right part of Figure 4.4. The solutions are depicted in the objective space as empty circles. It is possible to see that the result is strongly dependent on the values of weight vector.

4.3.2 Epsilon Constraint Approach

Another approach to transform the multi-objective problem into a single objective one is called ϵ-constraint approach. The main idea is to choose only one objective function to be optimized. Other objective functions are transformed into the constraints. For each of these transformed objective functions, the value ϵ_m is defined. This value is upper bound for the objective. If the value of objective function is greater than ϵ_m, the solution is considered infeasible. The transformed problem is defined as:

$$\begin{aligned} \text{minimize: } & f_\mu(\vec{x}), & & \\ \text{subject to: } & f_m(\vec{x}) \leq \epsilon_m & & (m = 1, \ldots, M), m \neq \mu \\ & x_i^{(L)} \leq x_i \leq x_i^{(U)}, & & m = 1, 2, ..., M \\ & g_j(\vec{x}) \geq 0, & & j = 1, 2, ..., J \\ & h_k(\vec{x}) = 0, & & k = 1, 2, ..., K. \end{aligned} \tag{4.9}$$

Here, f_μ is a chosen optimized function. Other objective functions are transformed into constraints in the form $f_\mu(\vec{x}) < \epsilon$. Other parts of the definition are identical with 4.1. Rule 4.3 provides an important property regarding the Pareto-optimality of obtained solutions [21].

Rule 4.3 The globally optimal solution of the ϵ-constraint problem stated in equation 4.9 is Pareto-optimal for any given vector $\epsilon = (\epsilon_1, \ldots, \epsilon_{\mu-1}, \ldots, \epsilon_{\mu+1}, \epsilon_M)$.

Figure 4.5 demonstrates the epsilon constraint transformation. The given optimization problem has two objectives and is similar to the demonstration of weighted sum approach. The optimized objective functions are given by equations 4.7 and 4.8. The second objective function f_2 is optimized f_μ, i. e. $f_\mu = f_2$ and the first objective function f_1 is transformed into the constraint. The results for three different ϵ values are shown ($\epsilon_1 = 0.1$, $\epsilon_2 = 1$ and $\epsilon_3 = 2.5$). The constraints are depicted using vertical lines and obtained solutions are marked by black circles.

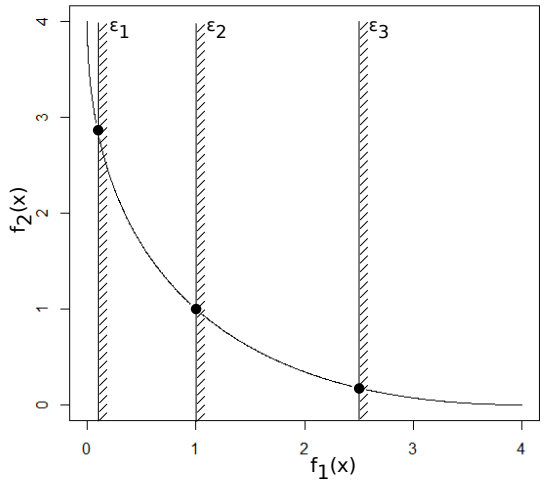

Figure 4.5: Epsilon Constraint Approach.

4.4 Genetic algorithm

The first version of GA was proposed by John Holland in seventies. GA is inspired by evolutionary and genetic processes of nature. GA is an iterative algorithm, which internally works with a set of candidate solutions. A new set of new candidate solutions replaces the previous set in each iteration. The new set of candidate solutions is produced using selection, crossover and mutation operators. In the area of GA, we call the iteration as generation and the set of candidate solutions as population. The items in the population are called individuals or chromosomes.

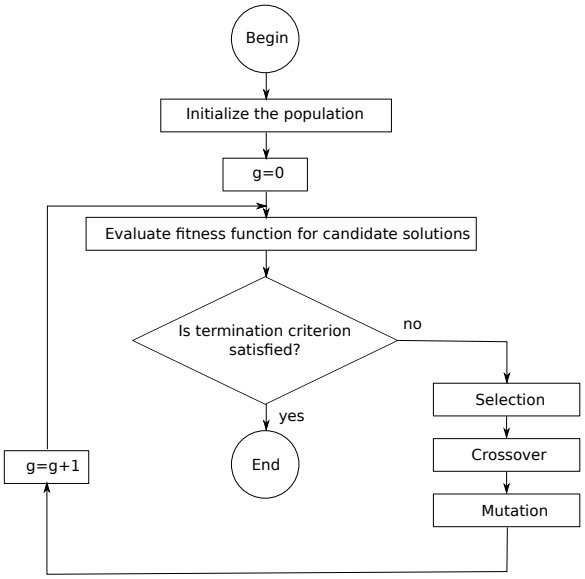

Figure 4.6: The principle of basic genetic algorithm.

GA works as follows (Figure 4.6). At the beginning the initial population is created. Fitness values are calculated for each individual in the population. Then the termination criterion is evaluated. If the termination condition is satisfied, the algorithm ends. Otherwise, the selection operator is performed. This operator selects perspective candidate solutions for the crossover and mutations. The crossover operator creates a new solution by using parts of two or more previous solutions. The mutation operator creates a new chromosome by a small modification of the parent chromosome [111].

4.4.1 Initial population

GA starts with creating the initial population. From the practical point of view there are two questions. How many individuals the initial population should contain and how to choose their parameters. In the case in which we choose a small population, we do not provide a sufficient room for exploring the search space effectively. On the other hand, if we choose a big population its processing can be very time consuming. In practice the optimal size of the population is often set after many experiments with the given problem.

The individuals are very often created using pseudo random generators. The pseudo random generator is executed for each item in the chromosome of a candidate solution. However many methods for more sophisticated generating of new solutions were proposed. For example, some kind of sampling can be used for better coverage of the search space. In

some cases the population is "seeded" using the solutions obtained by another optimization method. The solution is then further optimized using the genetic algorithm [39].

4.4.2 Termination condition

In general the genetic algorithm doesn't guarantee finding a global optimum. Manytimes it is also hard to detect if the global optimum was found. The most straightforward termination conditions is to stop the GA after consuming a predefined number of generations or predefined processor time. More sophisticated methods can be based on measuring the population diversity. If the population diversity decreases to a predefined threshold, the GA terminates. The diversity of population can be observed at the genotype or phenotype level. Sometimes these kinds of termination conditions are combined together in more complex conditions [39].

4.4.3 Selection

The goal of the selection operator is to choose perspective solutions for next population. The crossover and mutation operators are then applied on selected solutions. New solutions obtained by the crossover and mutation operators form a new population in the next generation of GA. In the proportional selection, also known as "roulette wheel", the probability r_i of selection of the solution is proportional to its fitness value f_i, i.e.

$$r_i = \frac{f_i}{\sum_{j=1}^{N} f_j} \tag{4.10}$$

where N is the population size. A random number u_i from uniform distribution within the interval $[0, 1]$ is generated. According to this random number (i.e. if $r_i \geq u_i$) the corresponding solution is selected.

It is sometimes too expensive to calculate the probability of selection for each solution in the case of the proportional selection. Also many arithmetic problems can appear, especially in the situation when values of fitness functions are very different. An approach, which is able to avoid these arithmetic difficulties is called the tournament selection. In each iteration of tournament selection, k (k is usually 2) candidate solutions are randomly chosen from the population. The fitness values of these solutions are compared and the solution with the best fitness value is selected. The process is repeated until a desired number of solutions is obtained [39].

4.4.4 Crossover

The crossover operator is able to create a new candidate solution by combining two or more previously found solutions. This operator is different for each type of chromosome. For binary chromosomes one point crossover and uniform crossover are often used. The principles of these two types of crossover are shown in Figure 4.7.

The one point crossover needs two parent chromosomes p_1 and p_2 and produces two children chromosomes c_1 and c_2. At the beginning, it generates a random number from 1 to the size of the chromosome. This number defines the splitting point. Both chromosomes are split into the two parts using this point. These parts are used to create new children. The child c_1 consists of the first part of p_1 and the second part of p_2. The child c_2 consists of the first part of p_2 and the second part of p_1.

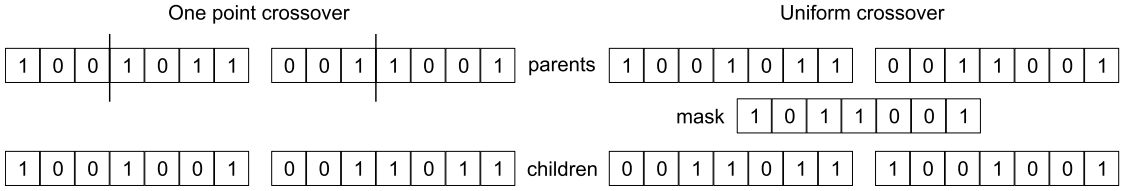

Figure 4.7: Binary crossover.

The uniform crossover also needs two parent chromosomes p_1 and p_2 and produces two children chromosomes c_1 and c_2. At first the random mask with the same size as parent chromosomes is generated. This mask contains only values zero and one and determines which parent will provide what item of the chromosome in the children, see Figure 4.7.

The Bernoulli distribution is often used to generate the mask. Bernoulli probability distribution takes value 1 with probability p and value 0 with probability $q = 1-p$ (equation 4.11). This probability distribution can be used, for example, to simulate toss of coin. In this case, the value of p is equal to 0.5 and represents the head. The tail is represented by the value q which in this situation, also equals to 0.5 [39].

$$x = \begin{cases} 0 & \text{with probability } p \\ 1 & \text{with probability } q \end{cases} \tag{4.11}$$

Both crossover operators described above are useful for binary chromosomes. In the case of chromosomes consisting of real numbers, other types of crossover should be used. One of them is called Simulated Binary Crossover (SBX) [22, 23]. The SBX uses the specially defined probability distribution $P(\beta_i)$ for generating of values in children chromosomes:

$$P(\beta_i) = \begin{cases} 0.5(\eta_c + 1)\beta_i^{\eta_c}, & \text{if } \beta_i \leq 1 \\ 0.5(\eta_c + 1)\frac{1}{\eta_c+2}, & \text{otherwise,} \end{cases} \tag{4.12}$$

where the parameter η_c is a non-negative number. If η_c is large, the value in the child chromosome will be near the parent value. If η_c is small, the generated value will be more distant from the value in the parent. β_i is called spread and is defined by:

$$\beta_i = \left| \frac{x_i^{(2,t+1)} - x_i^{(1,t+1)}}{x_i^{(2,t)} - x_i^{(1,t)}} \right| \tag{4.13}$$

Two probability density functions which differ in value of parameter η_c are shown in Figure 4.8. The function with $\eta_c = 2$ is marked by dashed line and the function with $\eta_c = 5$ is marked by the solid line. The parent solutions are placed on points 1, 4 and depicted as empty circles.

The process of creating new solutions by SBX is as follows. At the beginning a random number $u_i \in [0, 1]$ is generated for each gene i. After that, the ordinate β_{q_i} is determined from the probability distribution P. The ordinate B_{q_i} is defined such that the area under the probability curve is equal to the randomly generated number u_i. Value of B_{q_i} can be calculated by equation 4.14.

$$\beta_{q_i} = \begin{cases} (2u_i)^{\frac{1}{\eta_c+1}}, & \text{if } u_i \leq 0.5 \\ \left(\frac{1}{2(1-u_i)}\right)^{\frac{1}{\eta_c+1}}, & \text{otherwise.} \end{cases} \tag{4.14}$$

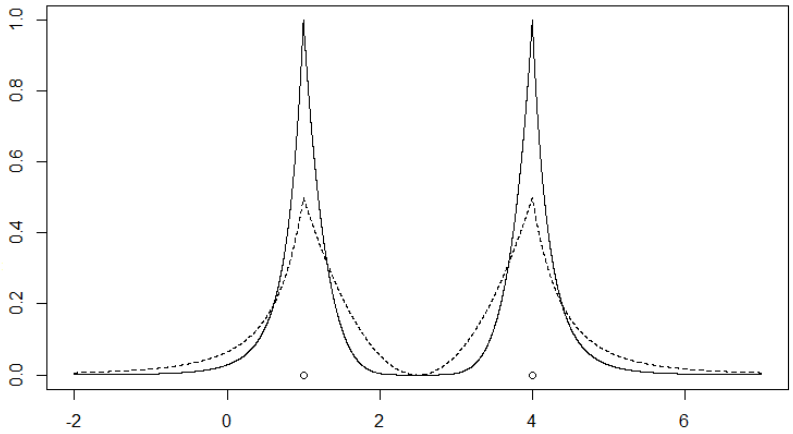

Figure 4.8: Probability density functions for SBX.

After it, the obtained ordinate B_{q_i} is used to create a new children. The values of gene i in children are calculated using equations 4.15 and 4.16.

$$x_i^{(1,t+1)} = 0.5\left((1+\beta_{q_i})x_i^{(1,t)} + (1-\beta_{q_i})x_i^{(2,t)}\right) \tag{4.15}$$

$$x_i^{(2,t+1)} = 0.5\left((1-\beta_{q_i})x_i^{(1,t)} + (1+\beta_{q_i})x_i^{(2,t)}\right) \tag{4.16}$$

4.4.5 Mutation

The mutation operator is capable of creating a new candidate solution by a small modification of the parent solution. In this thesis we utilize two types of mutation. The first one is designed for binary chromosomes and the second is designed for real valued chromosomes.

The binary mutation generates a binary value for each gene in chromosome. These values are generated using Bernoulli distribution (4.11) with a small value of p. A mask consisting of these numbers is then used to decide which values in the chromosome will be changed. If the mask value for corresponding gene is equal to 1 then the value of this gene is changed. Otherwise, the value of gene is preserved. Example of binary mutation is depicted in Figure 4.9 [39].

parent	1	0	0	1	0	1	1
mask	0	1	0	1	0	0	0
child	1	1	0	0	0	1	1

Figure 4.9: Binary mutation.

However, in the case of real valued chromosomes, there exist an infinite amount of possibly new values for this gene. One of methods used in this case is normally distributed mutation. The new value of gene is x_i^{t+1} is computed as:

$$x_i^{(t+1)} = x_i^{(t)} + N(0, \sigma_i), \tag{4.17}$$

where $x_i(t)$ is the value of parent's gene i and $N(0, \sigma_i)$ is a number sampled from the normal distribution with center 0 and standard deviation σ [49].

4.5 Multiobjective genetic algorithms

4.5.1 Vector Evaluated Genetic Algorithm

The Vector Evaluated Genetic Algorithm (VEGA) is a simple modification of a single objective GA [113]. VEGA algorithm is described in Algorithm 4.1. In each generation the whole population is randomly split into M subpopulations of the same size. The number of subpopulations corresponds to the number of objective functions for the given problem. In each subpopulation P_i the candidate solutions are evaluated using ith objective function. Then the selection process is performed on each subpopulation separately. The selected individuals are then gathered in a common mating pool and the crossover and mutation operators are used to generate a new population.

Set $q = \frac{N}{M}$
while *termination condition is not satisfied* **do**
  for *each* $i \in \{1 \dots M\}$ **do**
    for *each* $j \in \{(1 + (i-1)q) \dots iq\}$ **do**
      Evaluate fitness $F(\vec{x_j}) = f_i(\vec{x_j})$
      Perform selection on all q solutions to create a mating pool P_i
    end
  end
  Combine all mating pools together $P = \cup_{i=1}^{M} P_i$
  Perform crossover and mutation on P to create a new population
end

Algorithm 4.1: Vector Evaluated Genetic Algorithm

The VEGA is easy to implement, because it is only a simple modification of GA. Another advantage is that the algorithm doesn't require much more additional operations. On the other hand, the only one objective function is evaluated for one solution at a time. Thus, the algorithm tends to find optimal solutions for each objective, but the ability to find good compromise solutions is weak.

4.5.2 Strength Pareto Evolutionary Algorithm

The VEGA algorithm uses only one objective function at time to evaluate solutions. More advanced multi-objective genetic algorithms utilize the Pareto dominance relation to compare solutions. One of these algorithms is called Strength Pareto Evolutionary Algorithm (SPEA) [150]. This algorithm is capable of assigning a scalar fitness value to a solution. This scalar fitness function is calculated using Pareto dominance relation. The non-dominated solutions are stored in the external population. This feature guarantees that the best obtained solutions can't be lost during the evolution. To obtain solutions widely spread along the Pareto front, clustering operation is used. The main loop of SPEA is written in Algorithm 4.2.

Create initial population P and empty external population P'
while *termination condition is not satisfied* **do**
  Copy non-dominated members of population P to external population P'
  Remove solutions in P' which are dominated by any other member of P'
  If the number of solutions in P' exceeds a permited number, prune P' using clustering
  Calculate the fitness for each individual in P and P'
  Select individuals from $P \cup P'$ into the matting pool
  Apply crossover and mutation to create a new population
end

Algorithm 4.2: Strength Pareto Evolutionary Algorithm

We will describe the fitness evaluation step and the external population prunning step in detail. In the first phase of fitness evaluation, the fitness values are assigned to the members of external population P'. Firstly the "strength" value s_i is calculated for each candidate solution $i \in P'$ as:

$$f_i = s_i = \frac{n}{N+1}, \tag{4.18}$$

where n_i is the number of solutions from P which are dominated by solution i and N is the size of population P. The fitness value of solutions in the external population is equal to the strength value ($s_i = f_i$). When the fitness values of the external population are assigned, the fitness values for population P can be calculated. The fitness value of each solution $j \in P$ is equal to the sum of strength values of solutions in external population P' which dominate j plus one, i.e.

$$f_j = 1 + \sum_{i, i \succ j} s_i. \tag{4.19}$$

After the fitness values are assigned for the solutions from populations P and P' the selection operator is used to create the matting pool. The solutions are selected into the matting pool from the union $P \cup P'$. The crossover and mutation operators are applied on solutions in the matting pool in order to create a new population. Then a new iteration of SPEA algorithm begins.

Figure 4.10 explains the fitness assignment process. The example shows the objective space of a maximization problem with two objective functions. The solutions from the population P are marked as circles and solutions from the population P' are marked as crosses. The solutions are labeled using letters $A - K$. At first the fitness values of external population solutions are calculated. For example, solution A dominates solutions D, E and H in population P. The number of solutions in P is 8. Thus, the $f_A = s_A = \frac{3}{8}$. The resulting fitness values are displayed near the solutions. Then the fitness values for members of population P are assigned. For example, solution E is dominated by two solutions from the external population (A and B). The fitness for solution E is calculated as $f_E = 1 + s_A + s_B = 1 + \frac{3}{8} + \frac{6}{8} = \frac{17}{8}$. The process is identical for all other solutions in population P.

In the case that the number of solutions in the external population exceeds a given number N', the external population has to be prunned by clustering algorithm. This algorithm is described by Algorithm 4.3. At the beginning, a set of clusters C is created.

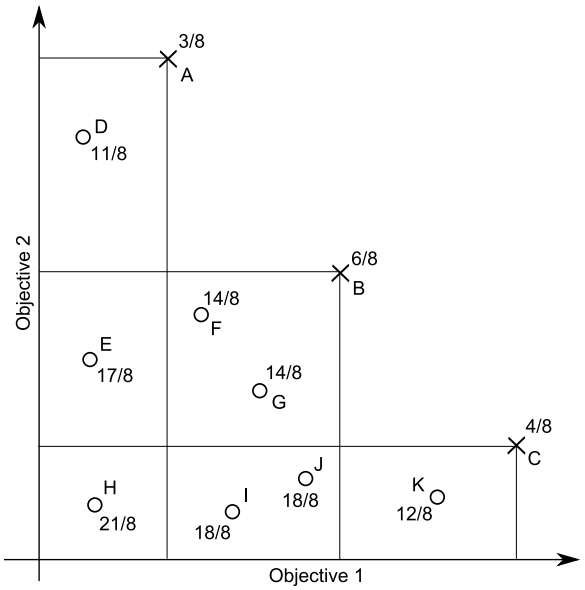

Figure 4.10: Fitness assignment in SPEA [150].

This set contains one cluster for each solution in the external population. Until the number of clusters in C is higher than N' the clustering loop is performed. The distance between each pair of clusters is calculated according to equation 4.20. Then, it is necessary to find two clusters with the minimal distance d. These two clusters are merged into one cluster.

$$d = \frac{1}{|c_1|.|c_2|} \sum_{i_1 \in c_1, i_2 \in c_2} ||i_1 - i_2||. \tag{4.20}$$

Inputs: External population P', which exceeds the permitted size N'
Outputs: Sub-population of P', with the permitted size N'
Create cluster set C. Each solution in external population is represented by one cluster.
while $|C| \leq N'$ **do**
  Calculate the distance for all pairs of clusters using equation 4.20.
  Find two clusters c_1 and c_2 with a minimal distance d.
  Merge clusters c_1 and c_2 ($C = C \setminus \{c_1, c_2\} \cup \{c_1 \cup c_2\}$).
end
Select a representative solution for each cluster. The centroid is used as a representative solution (The solution with a minimal average distance to other solutions in the cluster is used).

Algorithm 4.3: Clustering of external population in SPEA

When the clustering loop is finished the number of clusters is equal to the permitted number of solutions in the external population N'. In order to select the solutions which will sustain in the external population, the centroids of clusters are found. These centroids will sustain in the external population. Other candidate solutions will be removed from the external population. As the centroid of the cluster, the solution with the minimal average distance to other solutions within the cluster is taken.

4.5.3 NSGA II

The fast and elitist multi-objective genetic algorithm NSGAII splits the entire population into many non-dominated layers in each generation [26]. The solutions are selecterd into the new generation according to which non-dominated layer they belong to. A new approach for non-dominated sorting is utilized in the algorithm. This approach has the complexity $O(MN^2)$ instead of $O(MN^3)$ which was typical for previous methods, where M represents the number of objectives and N represents the population size. The algorithm also utilizes a mechanism for preserving of the diversity of candidate solutions. This mechanism doesn't need any sharing parameter like the previous version of NSGA. One of the important advantages of NSGAII is that it can easily deal with constraint optimization problems.

Inputs: Population P which contains candidate solutions.
Outputs: Sub-populations $F_1, \ldots, F_l$ which contain solutions from P according to non-dominated level.

```
for each p ∈ P do
    S_p = ∅ ;
    n_p = 0 ;
    for each q ∈ P do
        if p ≻ q then
            S_p = S_p ∪ {q} ;
        end
        if q ≻ p then
            n_p = n_p + 1 ;
        end
    end
    if n_p = 0 then
        p_rank = 1 ;
        F_1 = F_1 ∪ {p} ;
    end
end
i = 1 ;
while F_i ≠ ∅ do
    Q = ∅ ;
    for each p ∈ F_i do
        for q ∈ S_p do
            n_q = n_q − 1 ;
            if n_q = 0 then
                q_rank = i + 1 ;
                Q = Q ∪ {q} ;
            end
        end
    end
    i = i + 1 ;
    F_i = Q ;
end
```

Algorithm 4.4: Fast non-dominated sorting in NSGAII

The fast non-dominated sorting is described by Algorithm 4.4. At the beginning the algorithm finds for each solution p the domination count n_p and a set of solutions S_p that are dominated by solution p. Each p with $n_p = 0$ belongs to the first non-dominated front. Thus, the algorithm assigns the rank 1 for these solutions.

After the first non-dominated front is found, the counter i is set to 1. Until current non-dominated layer F_i is not empty, the following loop is performed. A temporary set Q is initialized to be empty. Then for each solution $p \in F_i$ and for each $q \in S_p$ the value of n_q is decremented. In the case $n_q = 0$ the rank of q_n is set to i and q is added into the set Q. After this process is performed, the counter i is incremented and F_i is equal to Q. Then a new iteration of the loop follows.

The fast non-dominated sorting algorithm is the central part of the whole NSGAII. At the beginning a random population P_0 is created and sorted using the fast non-dominated sorting algorithm. This procedure assigns a rank to each solution. Then the tournament selection is performed to select solutions for crossover and mutation. The solutions with a smaller value of the rank are considered as better. The algorithm performs crossover and mutation to create population Q_0.

When the populations P_0 and Q_0 are created, the main loop of NSGAII is executed. This loop is described by Algorithm 4.5. Both populations P_t and Q_t are merged into the set R_t. This set is sorted using fast non-dominated sorting algorithm. After the population R_t is sorted, the process of creating new population P_{t+1} begins. If the size of the first non-dominated subpopulation F_1 is smaller than the number of solutions in population N, the whole subpopulation F_1 is accommodated into P_{t+1}. The process continues until the new population P_{t+1} is not full. In the situation, the last subpopulation F_i can't be fully accommodated into P_{t+1} then F_i is accommodated only partially. The solutions of this subpopulation are sorted using a crowded comparison operator and only the solutions at the beginning are accommodated. Then the new population Q_{t+1} can be created using the operators of tournament selection, crossover and mutation from population P_{t+1}. The solutions are compared by the crowded comparison operator during the tournament selection.

while *Termination condition is not sattisfied* **do**
  $R_t = P_t \cup Q_t$;
  $F =$ perform fast non-dominate sorting algorithm on R_t ;
  $P_{t+1} = \emptyset$;
  $i = 1$;
  while $|P_{t+1}| + F_i \leq N$ **do**
    Perform crowding distance assignment on F_i ;
    $P_{t+1} = P_{t+1} \cup F_i$;
    $i = i + 1$;
  end
  Sort(F_i,$\succ_n$) ;
  $P_{t+1} = P_{t+1} \cup F_i[1 : (N - |P_{t+1}|)]$;
  Create new population Q_{t+1} using P_{t+1} ;
  $t = t + 1$;
end

Algorithm 4.5: NSGAII algorithm

The crowded comparison operator $\succ_n$ is defined by the equation:

$$a \succ_n b \text{ if } (a_{rank} < b_{rank}) \\ \text{or } ((a_{rank} = b_{rank}) \text{ and } (a_{distance} > b_{distance})). \tag{4.21}$$

This operator is used during the process of creating new population P_{t+1} and in the tournament selection. The operator is designed to prefer the solutions in less cowered regions of the Pareto front. At first the rank of two solutions is compared and the solution with the lower rank is considered as better. If the rank of both solutions is equal, the value of crowded distance is compared and the solution with the higher distance is considered as better.

In order to obtain solutions widely spread along the Pareto front, the NSGAII algorithm provides the mechanism to estimate the density of the solutions around the given solution. This mechanism is called crowding distance assignment and is described by Algorithm 4.6. At the beginning the value of crowding distance is set to zero for each solution within the given set I. Then for each objective m the following process is done. The solutions in I are sorted according to the objective m. The value of crowding distance for the solution with the smallest and the largest value of m is set to infinity. For other solutions, the value of the absolute normalized distance of two adjacent solutions is added to their crowding distance.

Inputs: A set of candidate solutions I.
Outputs: Value of *distance* for each item in I.
$l = |I|$ **for** *each* i **do**
  $I[i]_{distance} = 0$
end
for *each objective* m **do**
  Sort I using objective m $I[1]_{distance} = I[l] = \infty$
  for $i = 2 \dots (l-1)$ **do**
    $I[i]_{distance} = I[i]_{distance} + \frac{I[i+1]_{distance}(m) - I[i-1]_{distance}(m)}{f_m^{max} - f_m^{min}}$
  end
end

Algorithm 4.6: Crowding distance assignment in NSGAII

Example of the accommodation process is depicted in Figure 4.11. The whole box represents the union $P_t \cup Q_t$. The solutions in this union are sorted into subsets $F_1 \dots F_6$ using the fast non-dominated sorting algorithm. The first two subsets are fully accommodated into the new population. The subset $F3$ is too large to be fully accommodated. Because of it, the solutions with the larger values of crowding distance are preferred for accommodation. The remaining solutions of F_3 and solutions from subsets $F4 \dots F6$ are deleted.

In the past, various modification of NSGAII were proposed. For some problems, the strict elitism of NSGAII leads to premature convergence. In order to solve these problems, a new variant of NSGAII with controlled elitism was proposed [24]. Unfortunately, the original NSGAII is not too successful in solving multi-objective problems with a higher number of objectives (five and more objectives). These cases are often called as a many objective optimization problems. The new version of NSGA, called NSGAIII was proposed to solve this kind of problems [25, 59].

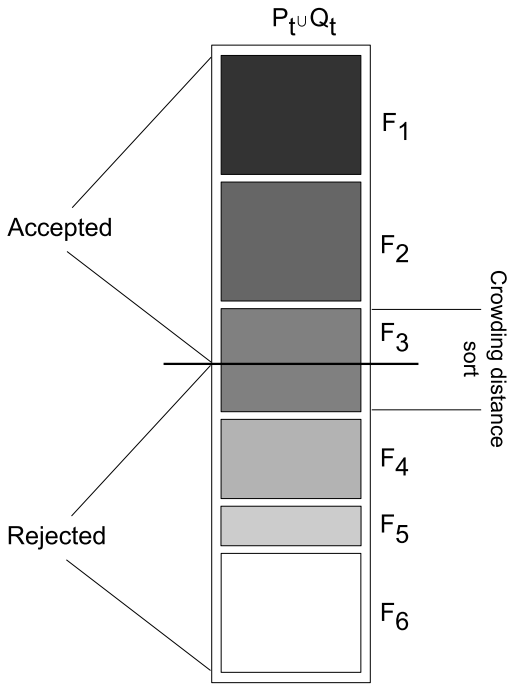

Figure 4.11: The mechanism of solutions accomodation into the new generation in NSGAII.

4.5.4 Multimodal NSGAII

In the multi-objective optimization different solutions often appear which have the same values of objective functions. In prediction tasks these solutions can, for example, represent different models with the same quality, but with different input sensors. Obtaining many solutions of this type can be very useful, because we can use the solution with available sensors and without any loss of quality. More formally, the multi-modal solution in the context of the multi-objective optimization is defined as follows:

Definition 3 *Two different solutions* $\vec{x}_1$ *and* $\vec{x}_2$ *(*$\vec{x}_1 \neq \vec{x}_2$*) are multimodal, if they have the same values of all objective functions (*$\forall i \in \{1 \dots N\}, f_i(\vec{x}_1) = f_i(\vec{x}_2)$*).*

To find multi-modal solutions, it is necessary to use modified versions of multi-objective genetic algorithms. In the past, a modified version of NSGAII for solving of multi-modal problems was proposed [27]. The difference between the standard NSGAII and modified version is in the accommodation of solutions into the new population.

In the modified (multimodal) version, all duplicate solutions are deleted from each non-dominated set $F_1, \dots, F_n$. Then the process of solution accommodation begins. It works as in the original NSGAII until it reaches the last set which can be accommodated. To decide, which solutions will be accommodated, we need to know the number of distinct objective solutions (n_l) and the number of non-duplicate solutions (N_l) in the last accommodated non-dominated set. The distinct objective solutions are defined by as:

Definition 4 *Two solutions* $\vec{x}_1$ *and* $\vec{x}_2$ *are distinct objective solutions if* $f_i(\vec{x}_1) \neq f_i(\vec{x}_2)$ *for at least one* $i \in \{1 \dots N\}$*.*

If the number of distinct objective solutions is higher or equal to the number of solutions to be accommodated ($n_l \geq N'$), then the standard NSGAII crowding distance procedure

is used to decide which solutions will be accommodated. Otherwise ($n_l < N'$) the procedure is different with respect to the original NSGAII. If the number of non-duplicated solutions N_l is higher than remaining population slots N', then the algorithm will at first accommodate one copy of distinct objective solutions. After that the proportionate rule is used to accommodate remaining solutions, in order to preserve approximately the same number of solutions for each distinct objective solution. Rarely a situation can appear, in which the number of remaining slots in the new population is higher than the number of non-duplicated solutions ($N_l < N'$). Then all non-duplicated solutions are accommodated and the remaining slots are filled by new randomly generated solutions.

Chapter 5

Road Traffic Flow Modeling

In this chapter, we will focus on the road traffic flow modeling. We will deal with the history of traffic modeling and traffic forecasting. Then the description of the basic relations between macroscopic traffic variables will be given. The general classification of traffic models with descriptions of basic approaches for traffic forecasting is also covered. Finaly, we will provide a review of soft-computing methods in short term traffic forecasting and describe some open problems in this area.

5.1 History of traffic flow modeling

The origins of traffic modeling and traffic flow theory dates back to 1930s, when Bruce D. Greenshields performed elementary observations of the traffic flow and provided a simple traffic model [44]. More intensive traffic research begun in the second half of the 20th century. In the 1950s, the computers were very rare and expensive. Most of them were used in the military domain. At this time it was not possible to use complex numeric or simulation models. The researchers focused mainly on theoretical models, which formed the base for the future traffic modeling. Many of these early models were inspired by fluid flow simulation (LWR theory) [85, 108]. These theories formed the principles of current macroscopic models. Another area studied in this decade was the car following, which provided the base for modern microscopic simulations [16, 48, 38].

The first traffic simulations appeared in the 1960s, but the simulation models were very simple. For example, TRANS model divided the road into cells. These cells can be occupied by vehicles, which hoped from one cell to another at a constant speed until the queue was reached. There was no car-following or lane changing logic [65]. More sophisticated model was UTCS-1, which used the car following and lane changing logic and the driver behavior model was non-deterministic [84].

In the next decade, there was a significant improvement in the traffic flow theory. The LWR model was improved and used for macroscopic traffic simulations [103]. The UTCS-1 model was enriched for vehicle emissions and fuel consumption [83]. Another model, called Texas, was introduced which provided detailed microscopic simulation, mainly designed to study safety aspects of the intersections [76].

Many new traffic simulators were developed or rewritten for IBM PC, for example Netsim [115], ROADSIM [96], FRESIM in the microscopic domain and INTEGRATION [134] in the mesoscopic domain. The quality and calibration of these traffic models become more important [7]. Moreover, a model for overtaking behavior [130] and the behavior

car-following models were developed [43].

In the recent two decades, the boom of the Internet enabled computers to be directly connected to sensors and provide traffic modeling in the real time. Because of it, the traffic models become more data driven. At the macroscopic level, models for time series prediction and soft-computing models become more popular. At the microscopic level, cellular automata based models, capable of exploiting the modern parallel computers were developed. The calibration of microscopic models using the real world data become a crucial issue.

5.2 Relations between traffic variables

The relations between the basic traffic flow variables (which were defined in Chapter 2) can be visualised by the so-called fundamental diagrams. Thess diagrams consist of three plots, each of them visualises the relations between two variables on a given place. The fundamental diagrams are based on general rules predicting that with a high number of vehicles on the road the mean velocity decreases. And when the critical density is reached, the traffic flow becomes unstable. The fundamental diagram provides us with the values of this critical density and speed [67, 40]. An example can be found in Figure 5.1.

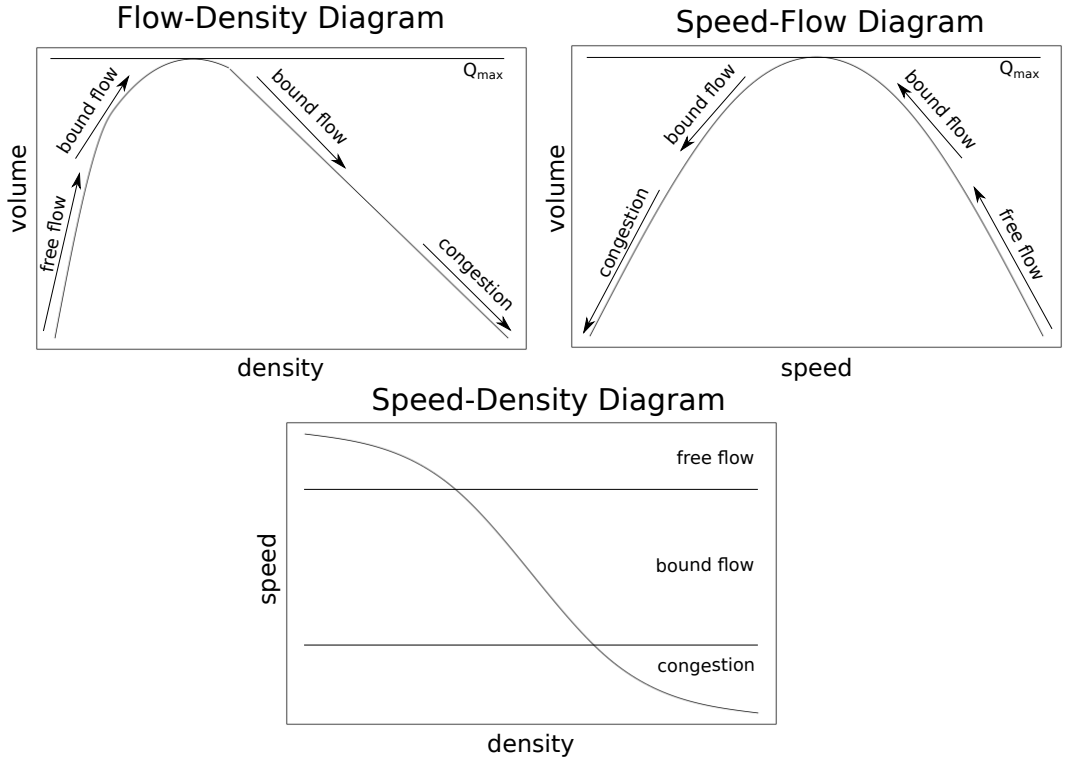

Figure 5.1: Fundamental diagrams

In the flow-density diagram, we plot the aggregated values of traffic flow against density. This diagram can be used as the basic source of information about the behavior of traffic in the given place. The road capacity is a maximal value of $Q(\rho)$. The actual speed can be calculated as the slope of the secant through points $(0, 0)$ and $(\rho, Q(\rho))$.

The aggregated values of speed against the aggregated values of volume are depicted in

the Speed-Flow diagram. We can identify the speed in which the maximal volume occurs, but the plot does not represent a function capable of computing the traffic volume from speed.

The speed-density diagram contains the speed values against the density values. The corresponding speed for the given density can be obtained from this figure. It is also possible to see that the value of speed decreases with increasing density.

5.3 Traffic prediction methods classification

In the past, various methods for short term traffic prediction were proposed. According to [135], we distinguish three basic groups of methods: naïve methods, parametric methods and non-parametric methods (Figure 5.2).

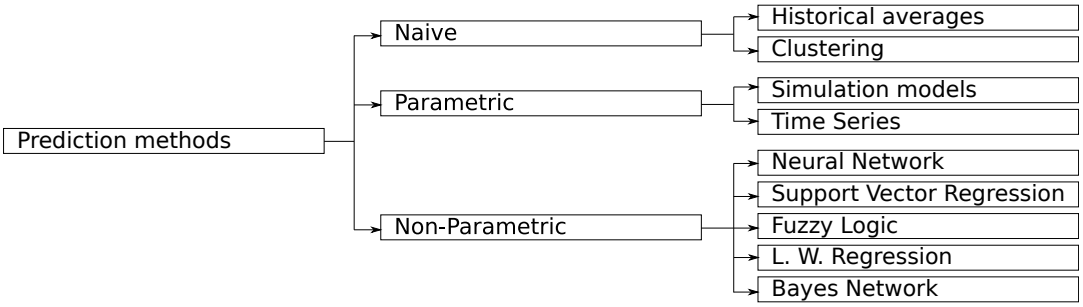

Figure 5.2: Short term traffic prediction models classification [135].

5.3.1 Naïve methods

The naïve methods have no model assumption. They are freequently used, because they are simple and can be easily implemented. However, the accuracy is usually lower in comparison with parametric and non-parametric methods. Many naïve methods utilize the historical data [98]. The average values are calculated from various parts of the day, combined with the last measured value and used as a prediction [50]. Other, more sophisticated naïve methods utilize clustering to distinguish different groups of days, according to the similarity of the traffic patterns [139].

5.3.2 Parametric methods

In parametric models, the structure of the model is pre-determined and only its parameters have to be found using available data. The main advantage of these models is that the amount of data to create a model is lower in comparison with the non-parametric models. Moreover, because these models are not as data driven as the naïve and non-parametric models, they can be successfully used to simulate traffic incidents and other unusual traffic situations.

Probably the biggest group of models belonging to this category are traffic simulation models. The main input data for them are the origin-destination matrices containing the number of trips between different locations in the area [8, 3, 9]. Traffic models can be divided into three subgroups, macroscopic, microscopic and mesoscopic models. The macroscopic models only consider the global traffic flow variables, such as intensity and occupancy.

They try to identify static equilibria in traffic flow. The dynamic macroscopic models are often inspired in hydrodynamics and try to predict the changes in traffic flows and traffic dynamics.

In contrast to the macroscopic models, the microscopic traffic models work on the level of individual vehicles. The agent based models consider each driver as an agent with his own state of mind and behavior. The model consists of many of these agents and captures the interactions between them [33]. Another microsimulation approach utilizes the cellular automata. In this approach, the road traffic network is divided into small cells and each cell can be occupied by a vehicle. In synchronous discrete moments, the state of cells is changed according to given rules simulating vehicle movement [97, 72].

There also exists a subgroup of traffic models called meso-scopic models, in which individual cars travel in the road network, according to macroscopic variables. An example of such approach is model called DynaMIT [6].

There is a group of parameter models considering traffic variables as time series and utilizing the common methods for time series modeling in order to predict the future values. The Autoregressive moving integrated average model (ARIMA) is capable to describe stochastic processes [45, 20]. However, it was discovered, that the traffic often shows a seasonal character. To deal with that, a modification called seasonal autoregressive integrated moving average (SARIMA) was proposed [123]. Athena is another example of the time series prediction model. This model uses a linear combination of previous values and the current value to predict the future values [69].

Another group of parametric models is based on the Kalman filter, which estimates the future state using previous and current state of the traffic [99, 146]. The Kalman filter can be also utilized in traffic data preprocessing and preprocessing of signals in various traffic sensors [71].

5.3.3 Non-parametric methods

In non-parametric methods, the structure of the model is not fixed and have to be learned from data. These methods can also have various meta-parameters, which are predefined by the user and which influence the overall structure of the model. The main drawback of these methods is that we need a relatively huge amount of data to train the model and this model should be then verified on testing data. The biggest advantage of these methods is that they can model difficult non-linear processes and they only require a minimal knowledge about the underlying process.

For example, the locally weighted regression belongs to this group. This is a memory based method which selects the nearest points to the point of interest from historic data. These historical points are then used to train a regression model, which performs the prediction [98]. Another model in this group is Bayesian network. In this model, the road network is represented in the form of a graph describing dependencies between traffic values measured on different roads [148].

Fuzzy logic [79] and machine learning algorithms such as neural networks or support vector regression also belong to this group. Because this thesis is mainly focused on machine learning methods, we have described them theoretically in chapter 3 and the summary how to use them in traffic forecasting is given in the remaining parts of this chapter.

5.4 Aspects of traffic forecasting

This section describes general aspects of traffic forecasting. These aspects include purpose for which we will perform the prediction; the aggregation aspect, which specifies whether we will work on the level of microscopic or macroscopic variables, and other aspects such as a mathematical structure employed and if this structure utilizes a random number generator [129, 52].

5.4.1 Purpose of traffic modeling

The area of traffic modeling covers traffic flow modeling and transportation planning. The traffic flow modeling has given the traffic demand and infrastructure. These two facts are fixed and the goal is to model the traffic under these circumstances. Contrary, the transportation planning models the dynamic of traffic demand effects of infrastructure changes.

From the temporal point of view, traffic flow modeling works with timescale of minutes to a few hours and transportation planning works with timescale from hours to years. The aspects of studied human behavior are also different. For traffic flow modeling, the important aspect is the operational driving behavior like accelerating, turning, lane-changing, or breaking. On the other hand, the transportation planning studies the number and type of trips and destinations. The scope of this thesis is mainly focused on traffic flow modeling.

The traffic flow prediction is mostly used for two purposes. The first one is the usage in advanced traffic management systems (ATMS), in which we try to utilize predicted variables to change the intersection signal plans. This can lead to more fluent traffic and better utilization of traffic infrastructure. The second purpose of traffic forecasting is in advanced travelers information systems (ATIS) that inform the drivers about the current and future state of traffic [90].

It is also necessary to take the area of deployment into consideration in the development of short term traffic forecasting methods. The algorithms for traffic forecasting usually differ for freeways and the urban areas. It is easier from this point of view to predict the traffic for freeways. The prediction for urban areas is usually more specialized and complex. The prediction in the highly congested urban areas is still a challenging task [119].

5.4.2 Aggregation level

According to the aggregation level, we divide traffic forecasting models to macroscopic and microscopic. In the macroscopic models, the traffic variables are locally aggregated. These locally aggregated variables are traffic density ρ, flow Q, mean speed V, speed variance and others. These models are mainly suitable for the description of collective phenomena such as future values of the aggregated traffic variable or estimation of travel times. The computational time of macroscopic models is usually quite short. This is especially desirable in the online traffic prediction, when the model must be evaluated in real time with limited resources. However, the macroscopic models do not consider many details, such as lane changes, individual driver behavior and others. To create a macroscopic model, we usually need only an aggregated data. This is advantage in the situation, when we have available data from various sources and it is necessary to perform some kind of data fusion before prediction [136].

The second group of traffic models are microscopic models. These models describe the state of the traffic using the collection of all vehicles in the area. The behavior of these

individual vehicles is evaluated separately according to the surrounding traffic. The typical microscopic dynamic variables are the vehicle position $x_\alpha(t)$, speed $v_\alpha(t)$ and acceleration $\dot{v}_\alpha(t)$. The most often used microscopic level models are the car-following model [43, 77] and cellular automata based model [97, 47]. It can be hard to find proper settings for the cellular automata to perform this task. It was shown that evolutionary algorithms are very suitable to find this setting [72, 73].

The main strength of the microscopic models is in the ability to capture the situations, in which the heterogeneity of traffic is important. For example, to consider different speed limits for various types of vehicles, bans on passing trucks and others. Microscopic models are capable to more precisely describe the human behavior such as accelerating, breaking, or lane changing. The microscopic models are also capable of predicting how a single vehicle can affect the traffic. This aspect is becoming more and more important for advanced driver assistance systems (ADAS) [41].

In between these two approaches, there exist few models which combine microscopic and macroscopic principle. These models are called mesoscopic models. For example, in parallel-hybrid models some parts of the traffic network are described microscopically and others macroscopically. The microscopic parts of models are usually used for intersections, traffic lights and other important parts. The other less complicated parts are described on macroscopic level [88].

5.4.3 Predicted variables

It is also important to describe what to predict and when. The most common approach is to predict three basic traffic variables, i.e traffic flow, occupancy and mean speed. There is still an open discussion, which traffic variable is the best indicator of traffic conditions. Some studies show that the prediction of traffic flow are more stable than of occupancy [78]. However, other authors prefer occupancy [87]. Additionally sometimes other traffic information can be the subject of prediction, for example, the queue length [75] in signalized intersections or onset of congestions [42].

When the predicted traffic variable is known, the next step is to define when the prediction will be performed. It is necessary to specify the prediction interval and prediction horizon. Both parameters have strong influence on the quality of forecasting. The forecast horizon is an extent time ahead to which the forecasting is made. Generally with increasing prediction horizon, the less accurate traffic forecasting becomes. However, the typical prediction horizon is 15 minutes or more [58]. The reason is that the shorter prediction horizon does not make a sense for users.

The time step is the time interval for which the forecasting is made. If the prediction interval is set too short, the prediction is usualy very poor because of large variability in the data. On the other hand, if the prediction interval is too long, the prediction can miss some important aspects of the traffic. Typical tasks of traffic forecasting have about a 5 minute time step and the prediction horizon of several steps ahead [36, 114].

5.4.4 Randomness

According to the randomness involved we distinguish deterministic models and stochastic models. Deterministic models don't use any random elements. The stochastic models use random elements to define some aspects of traffic flow. These aspects are modeled using the pseudo random generators [102].

The typical random elements are those which model unpredictability of a human behavior such as driver mood, acceleration or deceleration. A typical human driver sometimes accelerate or decelerate without any rational reason. The random elements can also incorporate some heterogeneity into the model. They can be used to define the vehicle types, parameters of vehicles, or weather changes. For example, cellular automata based models would not work correctly without these kinds of randomness [97].

5.5 Soft-computing methods in short term traffic prediction

In this chapter we will describe methods, which utilize neural networks or support vector regression to predict the future traffic state. The section is divided into two parts. In the first part, methods for forecasting of the basic traffic flow variables are described. The second part contains a description of methods for travel time estimation.

5.5.1 Traffic flow forecasting

Many studies dealing with neural networks in traffic forecasting were proposed in recent two decades. We will deal with those that are most connected to the topic of this thesis. One of the first studies published about the traffic flow forecasting by neural networks was written by Brian Smith in 1994. In his study, he predicted the value of volume in next 15 minutes by neural network with backpropagation learning. The inputs of neural network were the current volume, volume measured 15 minutes ago, historical volume and binary variable, which tells whether the pavement is wet. The results have shown that the neural network outperforms the historical average and ARIMA model during the peak periods [117]. Another study written in 1997 compares the time series methods, such as ARIMA and ATHENA with neural networks. The comparison was done on data from motorway near Beaune in France. However, the results of this study have shown that traditional time series methods outperform neural networks [69].

To successfully use neural networks, it is necessary to properly set the structure and various meta-parameters of the network. The current trend is to perform these tasks automatically. For example, genetic algorithms appear to perform well in this task. The evolutionary calibration of the neural network was tested by Vlahogianni et. al.. They proposed a method which can simultaneously optimize network dimension and learning parameters such as the learning rate or step size [137].

The first attempt to predict traffic flow using SVR appeared in 2002. The motivation for SVR was its good generalization ability for a limited number of training samples, rapid convergence and capability to avoid local optima during the learning process. The authors focus on prediction of traffic volume. The next value is predicted from a few previously measured values. The SVR is utilized to find the function in the following form:

$$y_{t+1} = f(y_t, y_{t-1}, \ldots, y_{t-n}), \tag{5.1}$$

where y_{t+1} represents predicted value and $y_t, y_{t-1}, \ldots, y_{t-n}$ are previously measured values. The method was tested using the data from one intersection in Xian city, but the authors do not provide any comparison with other methods [30].

Another method for short term volume prediction is based on Online-SVR, which is a SVR modification capable of learning continuously during the preduction phase [89]. This Online-SVR was tested to predict the traffic volume in typical and atypical traffic

conditions. The atypical traffic conditions are holiday traffic and the situation after the traffic collision. The authors compared the results of this method with other methods such as Gaussian maximum likelihood (GML), Holt exponential smoothing and neural networks. The results have shown that the OL-SVR has the second best prediction results after GML under typical traffic conditions and the best prediction results under the atypical traffic conditions [15].

Another approach is a combination of SVR with the chaotic simulated annealing optimization. In this approach, the SVR is utilized to perform the prediction, while simulated annealing optimizes the SVR meta-parameters. The prediction results have shown that this combined method outperforms seasonal autoregressive integrated moving average (SARIMA), Holt-Winters model and bacpropagation neural network [51].

SVR was also combined with a chaotic cloud particle swarm optimization. Similarly to the previous approach, the SVR is utilized to perform the prediction and the chaotic cloud particle swarm optimization optimizes the SVR meta-parameters. The method was evaluated using the traffic flow data from Dalian city [80].

5.5.2 Travel times forecasting

The neural networks appear to perform well in the task of travel time estimation. For example, Laurence Rilett proposed a method for forecasting freeway link travel time by multi-layer feed forward neural networks with the back-propagation learning algorithm. He considered four possible configurations of neural network inputs. In the first configuration, only the previously measured travel times on the road segment are used. In other three configurations, various subsets of neighbouring road segments were added among the inputs. The results for prediction 1 or 2 time steps ahead were best for the network, which used only the values measured on the given road segment. However, in longer prediction horizons, the neural networks with inputs from neighbouring sections performed better [101].

Another study utilized a different kind of neural networks called counter propagation network. The authors compared this neural network with traditional back-propagation and reported that the counter propagation neural network is one order of magnitude faster than learning of the backpropagation network and provides the results of the same quality [28].

SVR was used to predict travel times from the current and a few previously measured values. The method was evaluated by publicly available highway data from Intelligent Transportation Web Service Project [144, 145]. The authors compared SVR based method with other two methods. The first of them estimates the travel time from the speed at the enterance of the road sections using the following equation:

$$T(t, \Delta) = \sum_{i=0}^{L-1} \frac{x_{i+1} - x_i}{v(x_i, t - \Delta)}, \tag{5.2}$$

where Δ is a data delay, L represents the number of sections, $x_{i+1} - x_i$ denotes the length of the section and $v(x_i, t - \Delta)$ is the speed at the begining of the section. The second method predicts the future travel time as the mean value of travel time at the same time of day and the same day in the week according to the following equation:

$$\overline{T}(t) = \frac{1}{w} \sum_{i=1}^{w} T(i, t), \tag{5.3}$$

where w is the number of weeks considered in the prediction and $T(i,t)$ is the travel times value in time t in historic week i. The results showed that the SVR based method outperforms the above described methods [142].

5.6 Open problems

In this chapter we try to identify the open problems in the area of short term forecasting. A comprehensive summary of the open problems was provided in the article „Short-term traffic forecasting: Where we are and where we're going" which was written by Eleneni et al [138]. We will focus on those problems that correspond to the topic of this thesis.

Open problem 1: Arterial and network traffic predictions

Most short-term traffic forecasting algorithms were built to predict the freeway traffic flow. It is because the traffic prediction for city arterials is a much more complex problem. It is necessary to deal with new problems, such as signalization and traffic lights. The complexity of the problem also arises with the number of intersections and complexity of road network in which many roads may not be covered by measurement devices.

The data driven approaches, such as machine learning algorithms can succeed in this complex environment, where other conventional methods usually fail [104, 122, 31].

The methods proposed in this thesis are mainly designed to work in complex traffic networks. We used data from city arterials to evaluate the prediction quality of our methods.

Open problem 2: Short-term predictions: from volume to travel time

Most studies dealing with short term traffic forecasting focus on prediction of traffic variables such as volume and occupancy. It is mainly due to the traditional measurement devices such as radars and loop detectors that are able to measure these variables. However, in recent years, many devices capable to measure the travel times were developed.

As a part of this thesis, we proposed a new method to predict travel times. This method is able to combine the inputs from traffic sensors and modern license plate reading systems.

Open problem 3: Combining of the models

The quality of prediction is usually determined using an error metric such as the root mean squared error (RMSE) or mean absolute error. However, such a comparison is not always fair [64]. It is also necessary to consider other aspects, such as time complexity, adaptability, robustness and requested expertise and skills.

Because it is often hard to decide which model is the best one, it can be very useful to develop methods and heuristics which are capable of combining the results. For example, the approach which combines the backpropagation and RBF neural networks appears to outperform singular predictors on the task of freeway volume production. The Bayes rule was utilized to combine the results of two different neural networks in this approach [149]. Another method uses the neural network to combine the prediction results provided by three different models. These models are moving average, exponential smoothing and ARIMA [125]. The fuzzy logic also appears to work well in combining the model results [120].

In our approach, we combine multi-objective optimization methods with machine learning algorithms. These methods internally create different models, which are dynamically switched according to the currently available data.

Open problem 4: Employng the full potential of artificial intelligence

In the recent years, algorithms of modern artificial intelligence (AI) are becoming more widely used in the area of traffic systems. At the beginning the AI models were mainly used in the area of data analysis and traffic forecasting. However, it is possible to use them in many other areas such as modeling of driver behavior and employ them in decision making in modern ATIS and ATMS systems [93].

There is also a sceptical view, caused mainly by three reasons. Many results produced by artificial intelligence such predictors based on neural networks can not be interpreted by human and can be considered as a kind of black box. The second reason is that these methods do not guarantee finding an optimal solution. Moreover, many of them do not guarantee finding even a feasible solution. This problem is more visible for evolutionary optimization techniques. The third reason is that the AI methods often have various meta-parameters which must be set properly in order to obtain sufficient results. However, the setting of these parameters is often not trivial and requires a lot of expert knowledge. [18].

Methods designed in this thesis can calibrate their meta-parameters using multi-objective genetic algorithms or self-adaptation.

Chapter 6

Analysis of available traffic data

The traffic data are essential for understanding of the current traffic situation and traffic behavior. These data are obtained by methods described in Chapter 2 and sent by protocols such as Datex II or Alert-C into the traffic data centers, where they are stored, processed and further analyzed. In this chapter, we will discuss the data, which will be used for verification of our new methods for traffic prediction. Two groups of data are considered. The first group contains the data from the city of Seattle and we will use them to verify our methods for data imputation, traffic forecasting and travel time prediction. The second group contains the data from Prague and we will use them to verify our method for estimation of missing values in traffic density maps.

6.1 Research Data Exchange Project

The Research Data Exchange [2] is a project developed to share traffic data to researchers, application developers, and others. Provided data are well-documented and freely available to the public. The data are divided into many datasets according to the place of measurements or data source. Many different sources of measurements are available, for example traffic detectors or probe vehicles. Table 6.3 summarizes the available data sets.

Data environment	**Start**	**End**	**Sets**	**Size**
FDOT Orlando ITS World Congress	2010-09-01	2010-10-22	2	974 MB
Leesburg VA Vehicle Awareness Device	2012-10-18	2012-12-19	3	534 MB
Minnesota DOT Mobile Observation data	2013-06-26	2014-10-20	2	534 MB
NCAR 2009	2009-04-06	2009-04-22	8	826 MB
NCAR 2010	2010-01-28	2010-03-29	8	466 MB
Pasadena	2011-09-01	2011-10-31	37	348 GB
Portland	2011-09-15	2011-11-15	15	988 MB
Safety Pilot Model Deployment	2013-04-11	2013-04-11	5	2.4 GB
San Diego	2010-01-01	2010-12-31	14	24 GB
Seattle	2011-05-01	2011-10-31	12	24 GB
Vehicle Infrastructure Initiative	2008-08-21	2008-08-29	9	1.1 GB

Table 6.1: The summary of available data sets in Research Data Exchange project for year 2014.

This table contains the name of the data environment, the beginning and the end of measurement, the number of data sets and its size. From our point of view the most interesting data environments are those which contain the data obtained by traffic sensors or car license plate reading system, i.e. Pasadena, Portland, San Diego and Seattle data environment. For the experiments reported in this thesis we have chosen the Seattle environment because it contains the data from traffic sensors as well as from license plate reading system. Moreover, these data are measured in downtown and contains many missing values.

6.2 Seattle data environment

In this chapter, we will describe the Seattle data environment provided by the Research data exchange project. In particular, Seattle Sensys Dataset containing the data from sensors in the downtown of Seattle and Arterial Travel Time dataset containing travel time data are discused.

6.2.1 Available data sets

The Seattle data environment consists of the following data. The first data set is obtained by induction loops on I-5 freeway. The data are aggregated in 20 second intervals and consist of volume and occupancy measurements. The raw data are available in the data set called *20-second freeway data*. Because the 20 second intervals are too short for many tasks, the aggregation to five minute intervals is also available in the data set *Original 5-minute freeway data*. The volume and occupancy data can be supplemented by data about traffic incidents (*Incident Data* data set). Moreover, another information about travel times for I-5 freeway is given in the data set *Freeway travel times*. These travel times are calculated using the data collected from loop detectors using a vehicle trajectory algorithm. The data about freeway travel times are aggregated in five minute intervals.

The second part of the Seattle data environment consists of data measured using Sensys traffic detectors. These detectors are placed on intersections in the center of the city and provide the information about volume, occupancy and average speed. Another data set *Arterial Travel Times* contains the information about travel time of each single vehicle as recognized by a license plate reading system.

The rest of the Seattle data environment consists of a description of Seattle signals timing plans for arterials parallel to I-5 in *Seattle Timing Plans* data set, the scheduled and actual arrival times from King County Metro buses in *King County Transit AVL Data* data set, and the information about trains in *SeattleTransit Rail Data* data set.

6.2.2 Seattle Sensys Data

The Seattle Sensys Data was collected from traffic sensors. This data contains information about the traffic flow, occupancy and mean speed on selected intersections (see the map in Figure 6.1). In this case, sensors are placed at 23 intersections in the city. The data was measured from May 1 to October 31 and aggregated to 1 minute intervals. The dataset is split into three tables. The first table contains the unique number, GPS position and a short description of the place. The second table describes concrete sensors. Each row represents one sensor and contains the identifier of a given place and the sensor number. The position of a sensor at the intersection is defined by the lane identifier and direction. The data capturing technology and a short description of the sensor is also included. The

measured values are stored in the third table, where row contains the information about the volume, occupancy and average speed for the current minute on one sensor. In addition, a flag is stored which represents the quality of the measured values. There are three possible values of the flag. The value „0“ represents a correct measurement without any error. Values „1“ and „2“ are used to indicate bad measurements.

Column name	Range	Description
CabinetID	1-41	Unique ID of place.
SensorNum	1-24	ID number of detector on specific place.
Occupancy	0-100 (error: -1)	Percentage occupancy.
Volume	0-140 (error: -1)	Volume count.
Speed	0-99 (error: -1)	Speed in miles per hour.
Flag	0, 1, 2	Flags to indicate bad data (0=good).
HHMMSS	-	Local time.
YYYYMMDD	-	Current date.
GoodBit	0, 1	The value 1 indicates good data.

Table 6.2: The content of the table with sensors data.

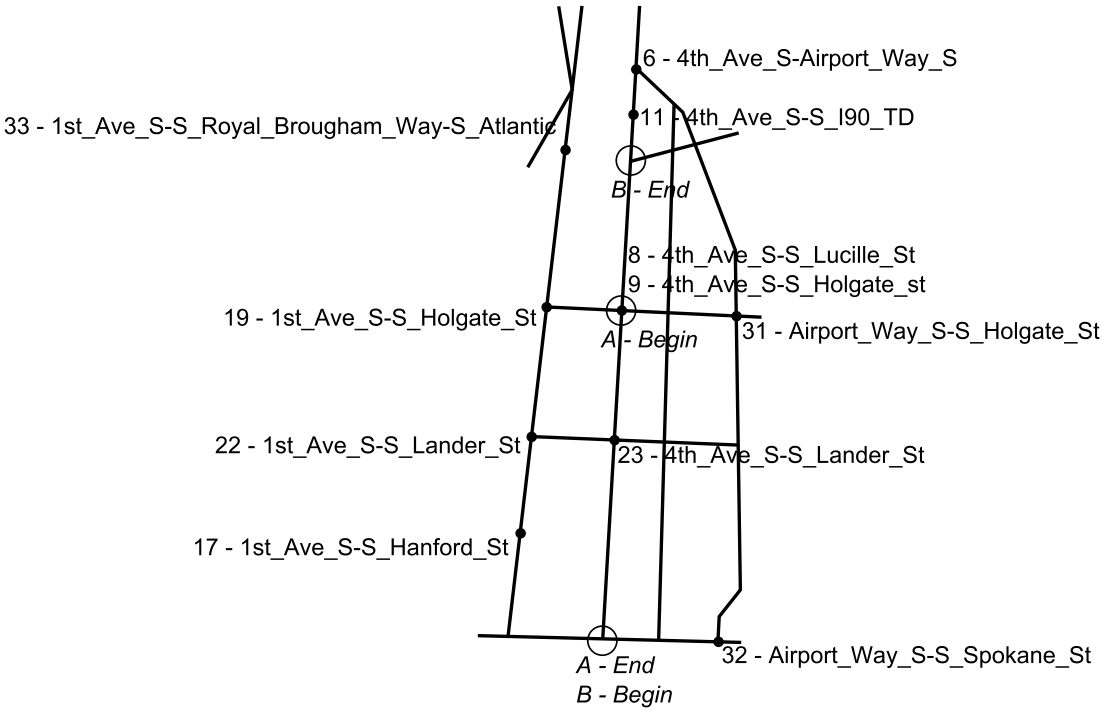

Figure 6.1: Map of the centre of the area. Sensors are marked by filled circles. Measured travel time road segments are marked by empty circles.

6.2.3 Data preprocessing

The raw data available on the Internet are aggregated in one minute time intervals. Our methods proposed in this thesis require and are evaluated on the data aggregated into 5 minute intervals. In order to provide descriptive data analyses, which works with the same

data as our methods we changed the aggregation level to 5 minute intervals. The statistics given in the following section are based on these transformed data. We used the sum as the aggregation function for traffic volume and the mean as the aggregation function for occupancy and speed.

6.2.4 Descriptive Analysis of Seattle Sensys Data

The first step to understand the given data is to perform a descriptive data summarization. In our case, we will mainly focus on the analysis of missing values, measuring the central tendency of data and measuring of the dispersion of data. The analysis of missing values gives us an information about how often and when the sensors are broken. It also indicates when it is necessary to use soft computing algorithms to predict these missing values, which is one of the motivations of this thesis. The analysis of central tendency shows the typical values measured on sensors. In the area of traffic data it is necessary to consider some common patterns in the traffic behavior. These traffic patterns are often dependent on time. The behavior of traffic variables can differ during the day. We can often see the morning and afternoon peak. The traffic behavior can be also different for day in week. We can see the huge differences between the typical values in working days and others. The third characteristic, which we will study for our data is the dispersion of data. This analysis provides the knowledge about the spread of the data and with the information about central tendency can help us to identify possible outliers in the data.

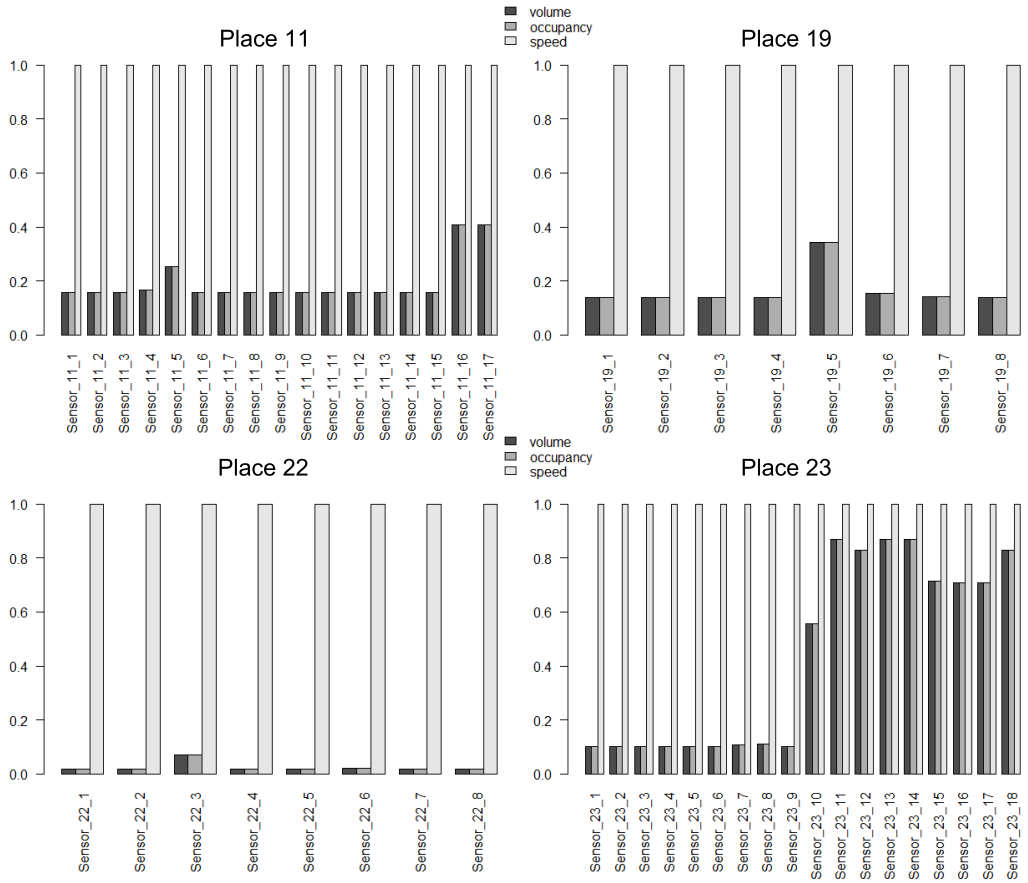

Figure 6.2: The rate of missing samples for all sensors on places 11, 19, 22 and 23.

The result of aggregation is given if there is at least one measured value within the time interval. For the analysis of missing values we selected four places (11, 19, 22, and 23) in the city center. Figure 6.2 shows the proportion of missing values for sensors located on the selected places.

It can be seen that the proportion of missing values for volume and occupancy usually stay bellow 20 percent. In some exceptional cases, the rate of missing values can be much higher, for example, for sensors 10-18 on place 23. Much different situation is for speed measurement. In the given data, there are nearly all values unavailable. It is probably because the sensors are located on crossovers and the cars very often have to stop there.

Another important issue is to discover if the rate of missing samples is dependent on the time in the day or day in the week. We chose sensor 4 on place 11 and sensor 4 on place 19 to analyze this behavior. The rate of missing samples according to day in week is depicted in Figure 6.3. As can be seen, there are no significant differences between the days in the week. Only Monday has a slightly higher rate of missing samples than other days in the week. The rate of missing samples according to hour in a day is depicted in Figure 6.4. The analysis shows that the differences are very small.

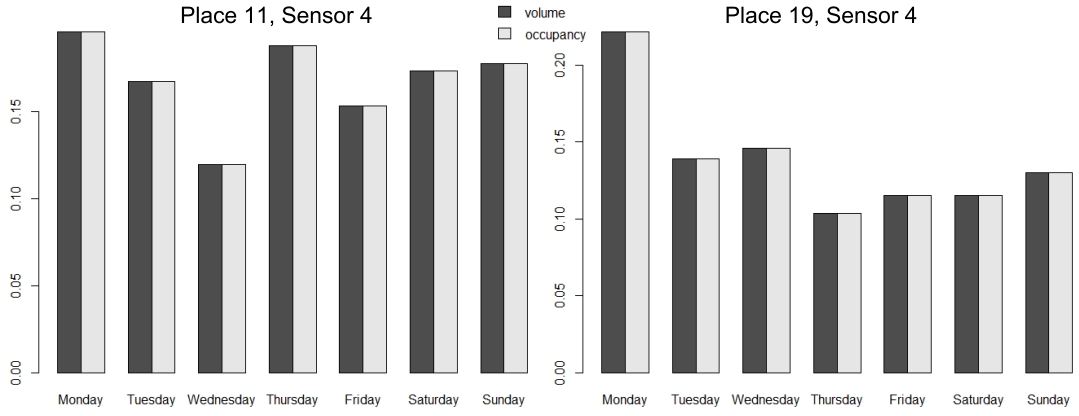

Figure 6.3: The rate of missing samples for sensor 4 on place 11 and sensor 4 on place 19 with respect to days in the week.

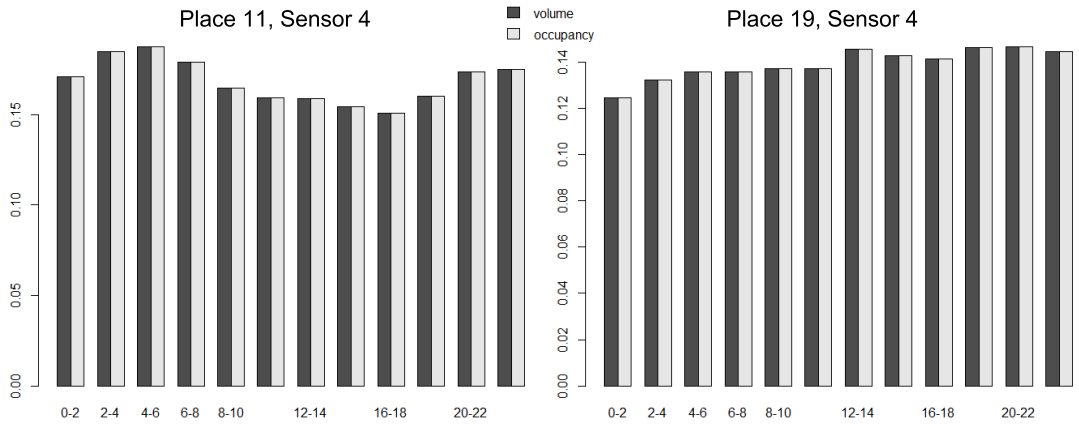

Figure 6.4: The rate of missing samples for sensor 4 on place 11 and sensor 4 on place 19 with respect to daytime.

In order to analyze central tendency of data we used the arithmetic mean

$$\bar{x} = \frac{\sum_{i=1}^{N} x_i}{N}, \tag{6.1}$$

where N is the number of values and x_i is the value of sample i.

However, the mean is not sometimes the best measure for central tendency of data. The main disadvantage is its sensitivity to outliers and extreme values. Even a single outlier can significantly change the value of mean. Many other measures like trimmed mean or median were proposed. The trimmed mean works like original mean, but some portions of extreme values are dropped. This portion is usually about 2-5 percent. If the portion of dropped data is huge (20 percent or more), the valuable information can be lost. On the other hand if we drop very small portion of data, there can still exist undesirable outliers and extreme values.

The variance and the standard deviation are the most often used measures for dispersion of data. The variance is as

$$\sigma^2 = \frac{1}{N} \sum_{i=1}^{N} (x_i - \bar{x})^2 \tag{6.2}$$

where N is the number of values, $\bar{x}$ is the arithmetic mean and x_i is the value of sample i. The standard deviation σ is equal to the square root of variance.

Figure 6.6 gives the overall summary of values measured on sensors. This figure is split into three parts. Each part contains the information about sensors, which measure a given variable. Each point in the figure represents one sensor. The horizontal axis represents the average value measured on the sensor and the vertical axis represents the standard deviation of measured values. The results show that with increasing the average value the standard deviation is also increasing. One can observe many missing values in speed measurements.

We also analyzed the dependency of measured values on daytime. In the case of occupancy and volume, there are usually two peaks. The first is at 9 o'clock, when people travel to the work and the second is at 17 o'clock, when they return back to home (Figure 6.5). The plots are in the form of boxplot and contain values for two sensors from 1st May to the 31st October. The data are aggregated into the five minute intervals.

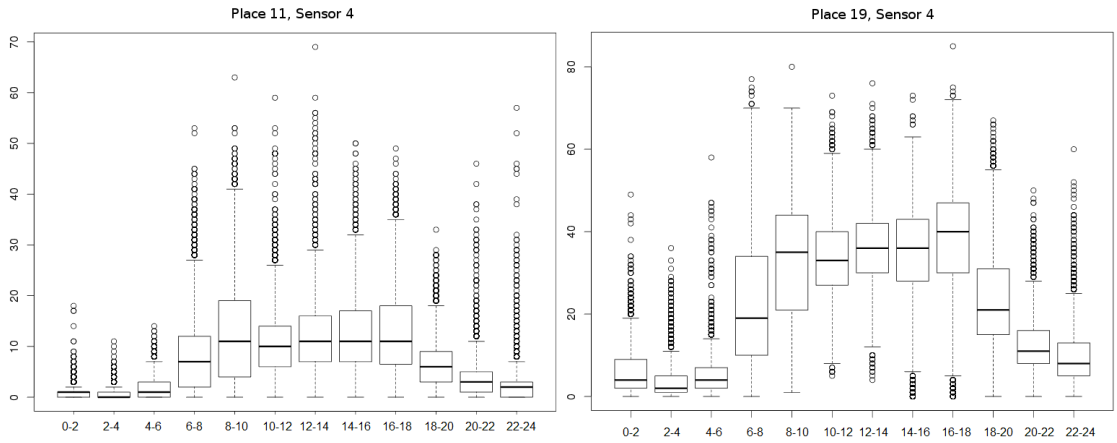

Figure 6.5: The dependency of measured volume on daytime. The data are visualised in the form of box plots.

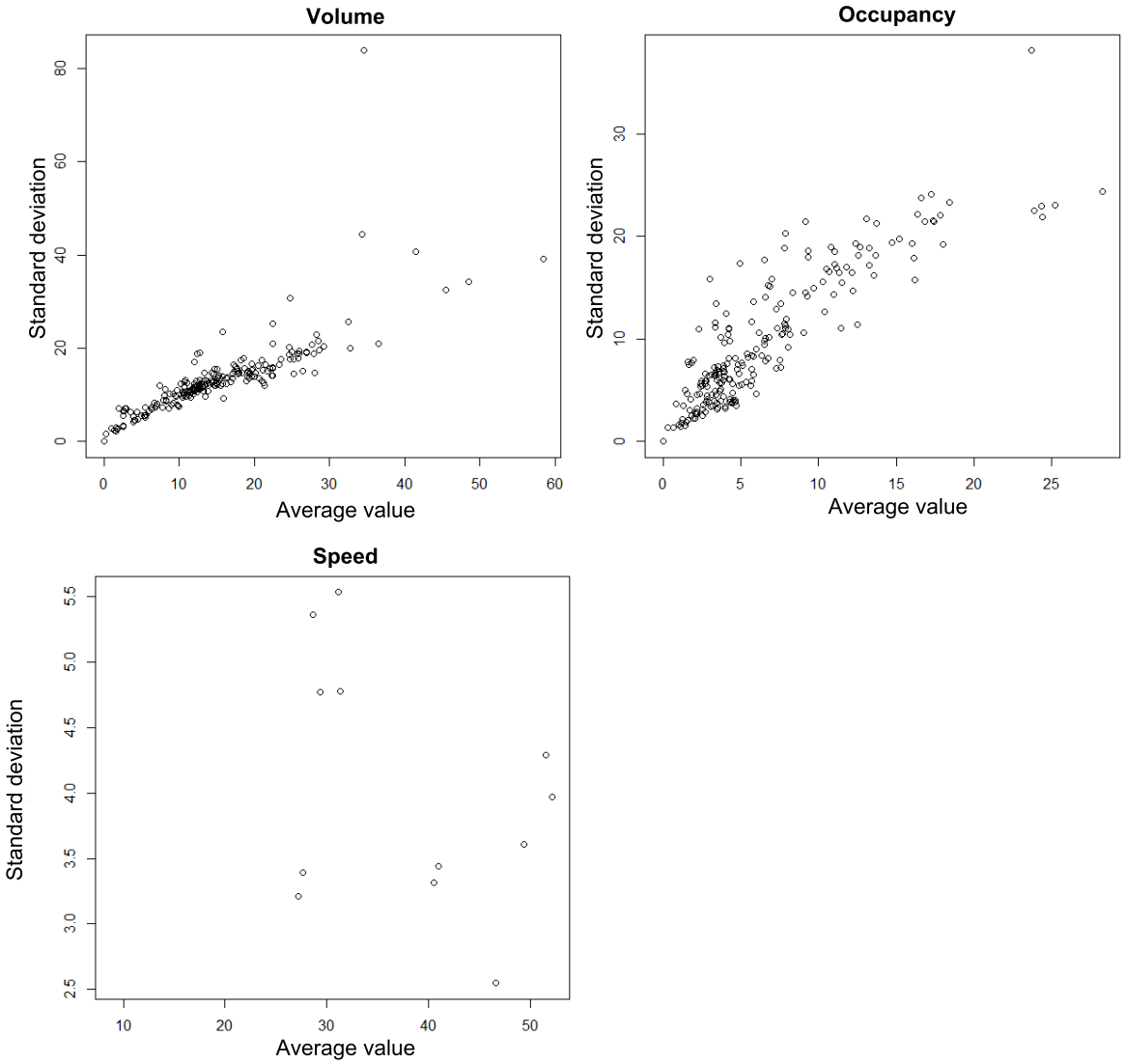

Figure 6.6: The mean and standard deviation of volume, occupancy and mean speed for sensors in the area.

Some degree of correlation between variables is desirable for machine learning algorithms. The correlation does not necessarily signalise the dependency. However, it is a valuable guideline. It can be also useful for an initial selection of machine learning algorithm's input. The Pearson coefficient is one of the most popular methods for calculation of the correlation (see 3.6).

To analyze the correlation between sensors, we calculated the Pearson coefficients for all sensors measuring the volume on places 8, 19 and 23. The values of correlation coefficient are shown in Figure 6.7 by color and the size of circle. The value close to +1, (or -1) indicates positive (negative) correlation. It can be seen, that correlations appear between many sensors, very often even between sensors located on different places.

Observation 6.1: The data obtained by traffic sensors contains many missing

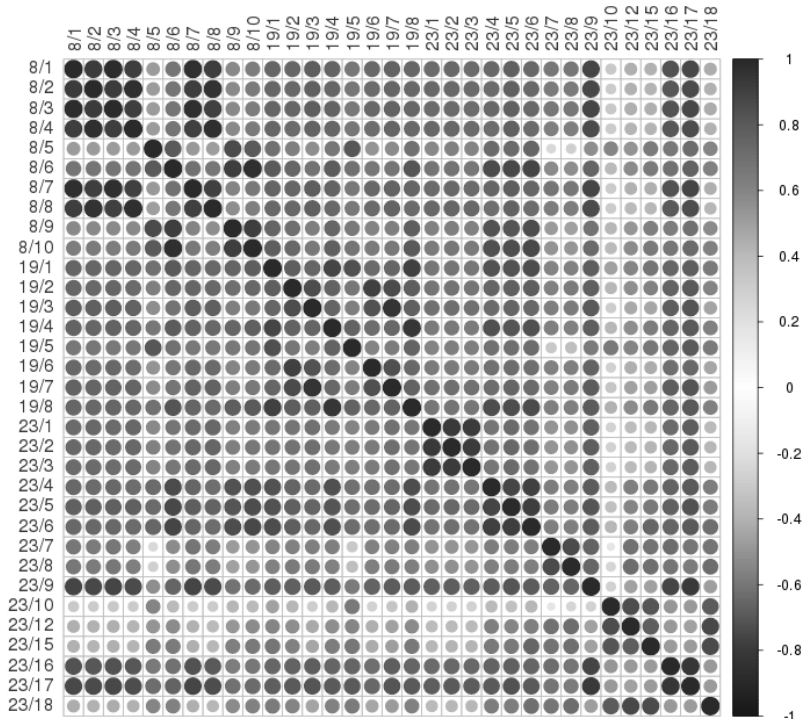

Figure 6.7: Sensor correlations calculated using Pearson coefficient. Each circle represents value of Pearson coefficient for two sensors.

values. The analysis has also shown that there are many correlations between sensors. It is very desirable to create a method capable to approximate the missing values using the data obtained by other sensors in the area.

6.2.5 Arterial Travel Times

The second type of data are the travel times of vehicles measured by a license plate recognition system. The data are distributed in two tables in *Arterial Travel Times* dataset. The first table provides locations of cameras, camera ID, GPS coordinates and primary and secondary streets. The second table contains the travel times of individual vehicles matched by the camera system (IDs of camera A and B, time stamps and travel time).

Column name	Range	Description
ID	1-1000000	A unique identifier of trip. The personal information are removed.
Begin Camera		The ID of the first camera.
End Camera		The ID of the last camera.
Date		Date of record.
Travel time		Car travel time in seconds.
Direction	N, S, W or E	Direction of the car travel.
Failed pass	0, 1 or 2	Data quality smoothing check.

Table 6.3: Content of the table with travel times data.

In the data, 97 different combinations of the begin and end places are recorded. This

means that there exists at least 97 possible trajectories of vehicles for which it is reasonable to measure travel times. However, most of these trajectories are not interesting for data analyses, because of the low number of vehicles recorder for them. We tried to analyse trajectories taken by more than 15000 vehicles per month. For this subset of trajectories we have calculated the median travel time and the number of passing vehicles per month. The results are visualised in Figure 6.8. The low number of passing vehicles is mainly due to errors during the license plate number recognition. It is important to note that the license plate number must be successfully recognized by the camera at the beginning and at the end of the trajectory.

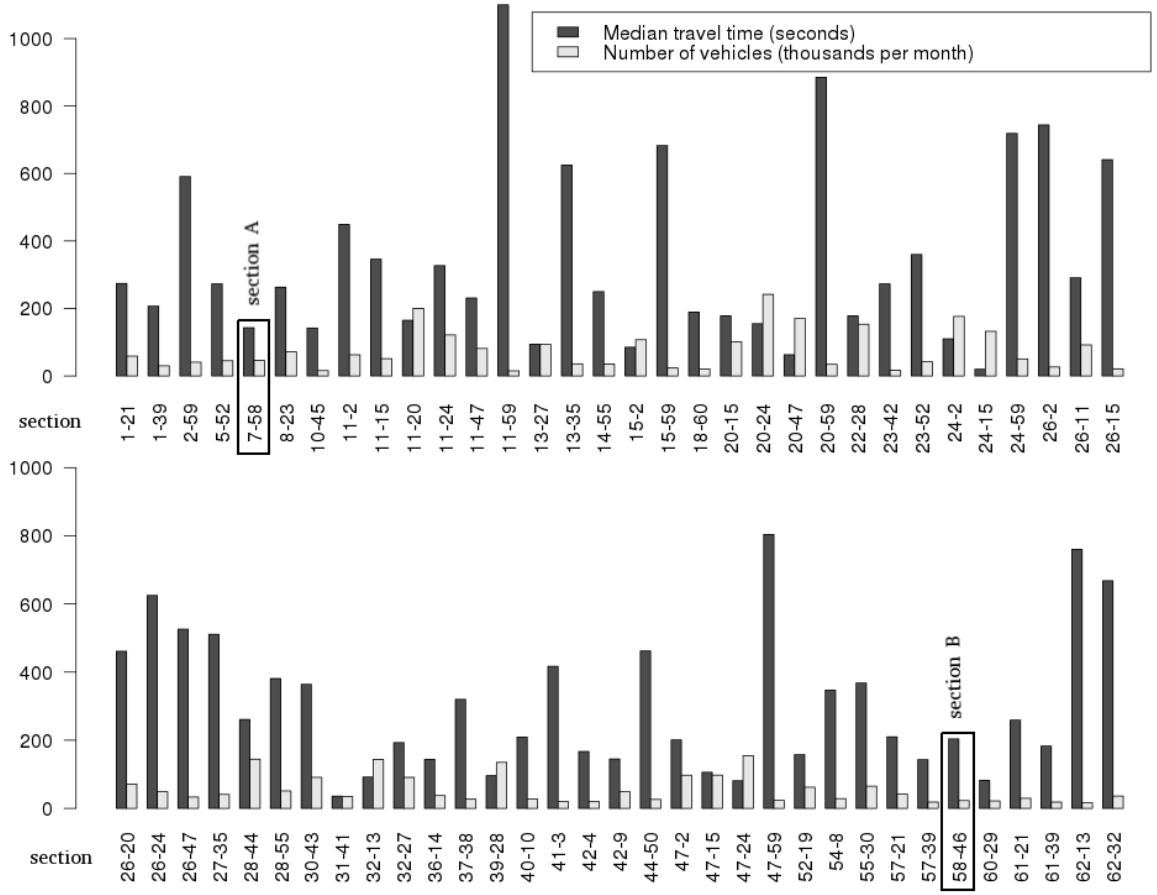

Figure 6.8: The median travel time and the number of vehicles for different trajectories.

Finally, we selected only two trajectories for further analysis. We denoted these trajectories as A and B. The map in Figure 6.1 shows them using the empty circles. The first camera of the trajectory A has number 7 and is located on the place 8 (according to sensys data). The second camera of the trajectory A has number 58. The trajectory B has cameras with numbers 58 and 46. These trajectories were chosen mainly because they are located in the centre of the area and are surrounded by many other traffic sensors. The number of matched vehicles is also not high for these trajectories. This can be also useful, because we want to show that our methods are capable to deal with missing data.

We analysed how the availability of data is changing during the day and the week. We provide the results for sections A and B in Figure 6.9. It is possible to see that the number of matched vehicles is significantly lower during the evening hours and at the night. This

can lead to worse travel time estimation during these periods. However, the importance of travel time information for drivers is also lower during the evening and night. It is important to note, that the number of detected vehicles is lower in the weekend.

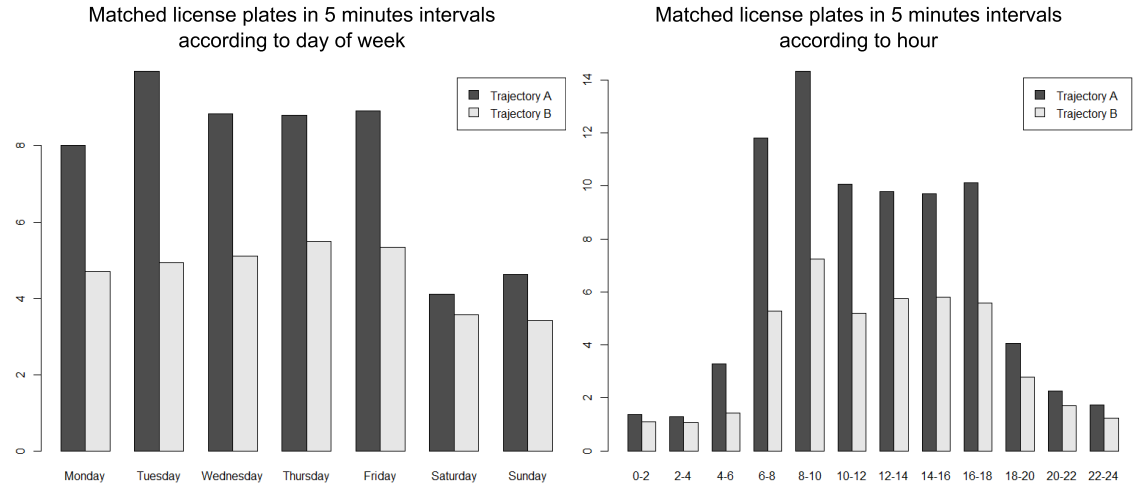

Figure 6.9: The mean number of matched license plates in five minute interval according to day of week and daytime.

The license plate data usually contain many outlier values. These values are caused by vehicles, which do not use the shortest path. The driver can also stop for some reason somewhere in the middle of the segment and after a break he/she continues. We tried to detect these outlier values for our sections, see Figure 6.10.

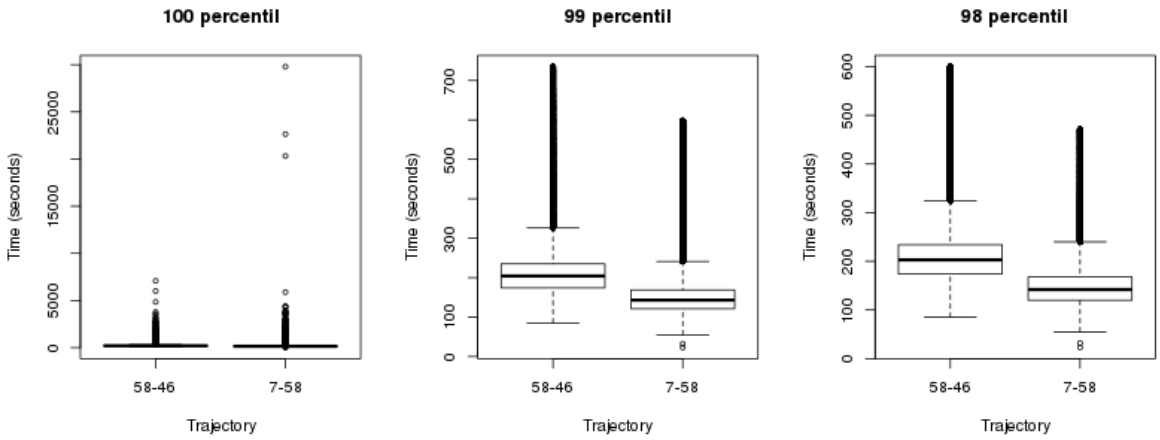

Figure 6.10: The values of vehicles travel time for trajectories A (7-58) and B (58-46). The values higher than the given percentiles are filtered out.

On the left side, there are shown unfiltered data in the form of boxplot. It is possible to see that many measured values are extremely high. The next plots show the situation after filtering the values higher than 99 and 98 percentile. It is possible to see on the right plot, that the unreasonably high travel times are not present. In the further analyses, we will use a license plate data smaller than 98 percentile.

We also try to figure out how the travel time depends on the hour of the day. The results for both sections A and B are shown in Figure 6.11. The longest travel times are the highest from 12:00 to 14:00 for the section A and from 16:00 to 18:00 for the section B.

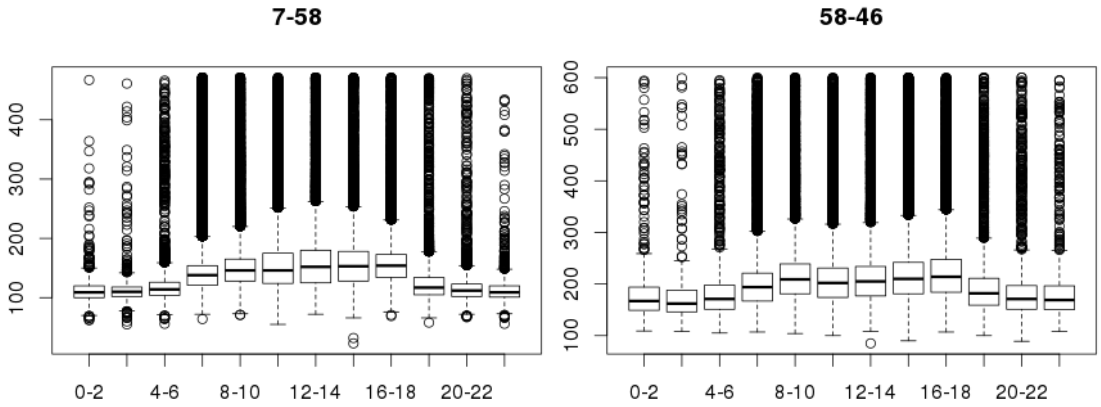

Figure 6.11: Vehicles travel times for trajectories A (7-58) and B (58-46). The values are measured for different hours in the day.

We used a correlation analysis to detect whether there can be some dependencies between travel times and values measured on traffic sensors. The results of this analysis for section 7-58 and volume sensors on two places (8 and 23) are depicted in Figure 6.12. In most cases, strong correlations can be observed.

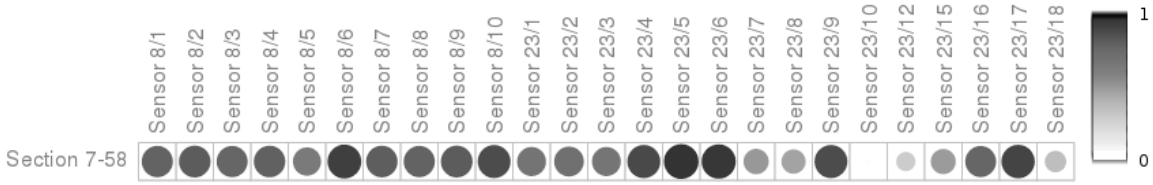

Figure 6.12: Sensor values versus travel time correlation.

Observation 6.2: The travel times data provided by the license plate reading system are strongly dependent on the daytime. The number of matched vehicles decreases at night and early in the morning. The travel times are also deteriorated by many outlier values. On the other hand, missing values in the data comming from traffic sensors do not depend on daytime and do not contain too many outliers, but it is harder to use them for travel time prediction. A method is needed capable of utilizing both data types (from license plate reading system and sensors) to provide more robust and precise travel times prediction.

6.3 Prague data

The second source of data for our work was the dataset for Prague. This dataset is not provided by ITS-RDE and we used it mainly to verify our method for estimation of missing values in traffic density maps.

6.3.1 Data description

The dataset contains the information about traffic volume in Prague for years 2008 and 2009 (Table 6.4). It is split into two tables. The first table provides the information about intersections (intersection id and GPS coordinates). The second table contains the data for road segments connecting the intersections. The id, beginning and end intersections define each segment. The table provides the value of intensity for segments measured in the year 2008 and 2009.

Number of intersections	126
Border intersections	28
Inner intersections	98
Number of road segments	277
Measured road segments in 2009	117
Not measured road segments in 2009	160

Table 6.4: Prague data - basic description.

6.3.2 Basic data characteristics

We analyzed the values of traffic volume in the year 2008 and 2009. It appears there are a few segments with very high volume (about 50000 vehicles) and many others with values about 10000 vehicles. The histogram of these values is depicted in Figure 6.13. Based on the Prague data, we postulated the following observation:

Observation 6.3: An algorithm with specific properties is needed to estimate missing values in traffic networks. In particular, the algorithm should be able to tolerate inaccurate measurements and it should be able to exploit data from previous measurement to improve the quality of the result.

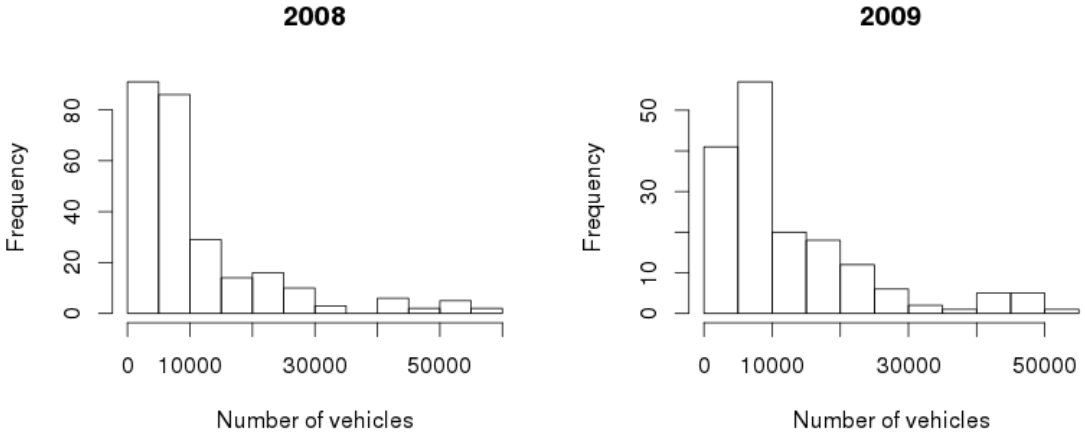

Figure 6.13: Histogram of measured values in the year 2008 and 2009.

Chapter 7

Estimation of missing values in traffic density maps

7.1 Traffic density maps

The traffic density map (TDM) represents the density of road network traffic as the number of vehicles per a specific time interval. This interval can be given in minutes or hours. Usually, TDMs are used by traffic experts as a base documentation for planning a new infrastructure (long-term) or by drivers for showing a current traffic status (short-term). Such TDMs can be composed automatically – with the aid from standard surveillance technologies (e.g. various data sensors such as loop detectors or traffic cameras). Another approach, which can be used for TDM calculation, is the manual counting on selected road segments. However, counting where people are involved in the process is usually quite inaccurate and also inefficient [100].

There is also a big effort to estimate the future traffic density. For example, paper [100] reports some techniques of density prediction for the congestion analysis under heterogeneous traffic conditions. The goal is to determine their feasibility under the Indian traffic scenario. Recently, a statistical approach to predict the density on any edge of such a network at some time in the future was also presented [74]. The method is based on short-time observations of the traffic history. In the two previous examples, however, complete data sets for the whole traffic network were required. The approaches require either a lot of traffic sensors or in the worst case, many people involved in the manual counting. In the situation where it is not possible to cover the whole traffic network with the field data, missing areas must be completely excluded from the traffic density estimation. In our work, we deal with a more realistic scenario in which TDM is not complete.

TDM can be viewed as a directed graph, where each node n represents a crossroad and each edge represents a particular road segment. The density on the edge, d_e, represents the number of incoming or outgoing vehicles per time interval on a given edge e. The historic value of density h_e (e.g. measured a year ago) can also be available for some edges. Nodes in TDMs are divided into two sets. In the first set, there are border nodes and we denote this set as N_b. All incomming and outgoing vehicles from the investigated traffic density map must go through these nodes. The second set contains the inner nodes. We denote this set N_i. For the inner nodes it also applies that the sum of the input vehicles minus the sum of the output vehicles should be near, or even better, equal to zero. Both sets are strictly disjunct ($N_i \cap N_b = \emptyset$). Every node n has a set of incoming edges I_n and a set of

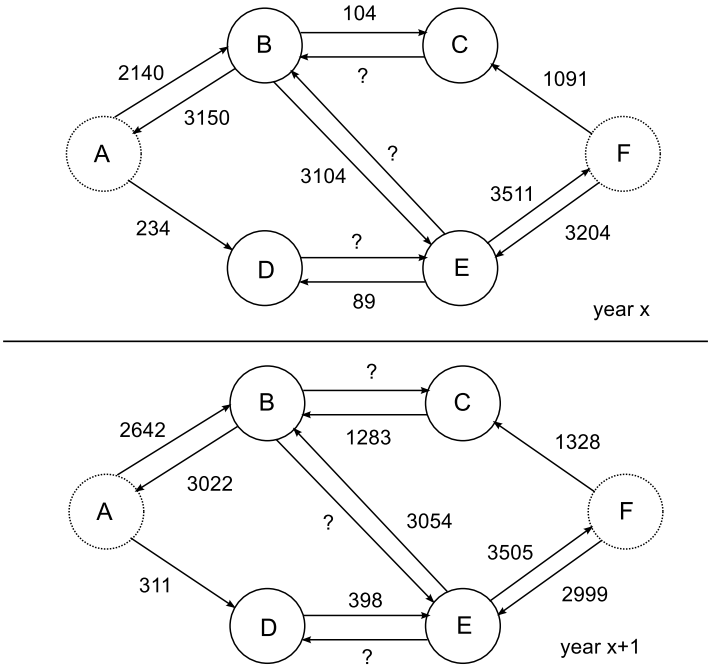

Figure 7.1: Syntetic example of measured data for TDM in two years. Missing values are marked by question mark. Values on these edges should be estimated.

outgoing edges O_n, respectivelly. Fig. 7.1 shows a TDM consisting of 6 nodes and 12 edges, where $N_b = \{A, F\}$ and $N_i = \{B, C, D, E\}$. In TDM which we consider in this work, one has to deal with hundreds of nodes and hundreds of edges. Typically 40 % of values are missing and have to be estimated.

7.2 Quadratic programming approach

Quadratic programming is a method for solving optimization problems that can be formulated in the following form 7.1.

$$\begin{aligned} &\text{Minimize } q(\vec{x}) = \frac{1}{2}\vec{x}^T G\vec{x} + \vec{x}^T\vec{d} \\ &\text{subject to } \vec{a_i}^T\vec{x} = \vec{b_i} && i \in \xi \\ &\qquad\qquad \vec{a_j}^T\vec{x} \geq \vec{b_j} && j \in I \end{aligned} \tag{7.1}$$

where $\vec{x}$ and $\vec{d}$ are vectors with n components and G is a positive semidefinite matrix. Expressions $\vec{a_i}^T\vec{x} \geq \vec{b_i}$ and $\vec{a_i}^T\vec{x} = \vec{b_i}$ represent some constraints [10], where ξ is set of equality constraints and I is the set of unequality constraints.

In order to solve a particular TDM problem, the quadratic programming is used to minimize an absolute value of the difference betweeen the number of incoming and the number of outgoing vehicles for all nodes of TDM. The error of an inner node E_n can be calculated using equation

$$E_n = \left| \sum_{e \in I_n} d_e - \sum_{e \in O_n} d_e \right|. \tag{7.2}$$

The squared sum of these errors for all those nodes is then our objective function

$$E = \sum_{n \in N_i} E_n^2. \tag{7.3}$$

In this approach, we can define a set of constraints for edges e which we don't know the density value d_e for. As we may know historic values from the previous measurement h_e, it is possible to constraint the expected value to be in some range (e.g. $d_e \in [0.7h_e, 1.3h_e]$). The exact ranges must be specified by a traffic expert.

The main advantage of this approach is in its speed. It's guaranted that the QP method will always find a solution. Our problem instances were solved in a few seconds. On the other hand, a quite strict problem formulation does not allow us to simply incorporate more domain knowledge into the quadratic programming method, for example, to define more objective functions for the optimization.

7.3 Estimation of missing values using multiobjective genetic algorithm

In this thesis, we propose a new approach for estimation of missing values in traffic density maps, which is based on NSGAII. This enables us to obtain more realistic solutions because we can consider more aspects in the optimization process.

7.3.1 Encoding of parameters and genetic operators

In our approach, each candidate solution is defined by a vector of real numbers. Every component of the vector represents a traffic density on one road segment, for which the density is not avalaible. The parameter value should be rounded to have the integer value. In the first generation of GA, components of vectors are initialized to positive randomly generated values $[0, 100000]$.

In the quadratic programming approach, it was possible to use only one objective function (see Eq. 7.3). The main reason to utilize NSGAII is that it allows us to use more fitness functions directly. In our case there will be two objective functions. The first is the sum of errors on nodes (see Eq. 7.4) and the second is the sum of differences to historic values (see Eq. 7.5).

Let E_h be a set of edges which have not the density values available, but we know the historic values of density for them. The second objective function F_2 is then defined by equation 7.5.

$$F_1 = E = \sum_{n \in N_i} E_n^2 = \sum_{n \in N_i} \left(\left| \sum_{e \in I_n} d_e - \sum_{e \in O_n} d_e \right|^2 \right) \tag{7.4}$$

$$F_2 = \sum_{e \in E_h} |d_e - h_e| \tag{7.5}$$

Similarly to the quadratic programming, it is possible in NSGAII to constraint the expected density to interval $[0.7h_e, 1.3h_e]$.

A single point crossover and a normally distributed mutation are utilized. The single-point crossover swaps some parameters between two candidate chromosomes. Normally distributed mutations work as follows. For each gene g_i of chromosome, a random number

$r_i \in [0, 1]$ with the uniform distribution is generated. If this number is less than the mutation probability P_t, then a new randomly generated number (with the normal distribution $N(0, \epsilon)$) is added to gene g_i.

7.3.2 Self adaptation

In order to maximize performance of the genetic algorithm it is necessary to correctly set various control parameters such as the population size, the probability of crossoever, the probability of mutation etc. This can be considered as an optimization problem itself.

These control parameters can be determined by expert, or discovered by another genetic algorithm (this approach is called metaevolution). In our evolutionary approach, we use a self-adaptive method, which enables to encode some control parameters of genetic algorithm into the chromosome [49], [4] and [35]. We included into the chromosome: a special gene g_σ to represent the deviation of mutations, gene g_c to represent the probability of crossover and gene g_m to represent the probability of mutation. Tab. 7.3.2. shows permited values and deviations for mutation of these genes.

Gene	Permitted values	Deviation
g_σ	$[0.00002, 0.2]$	0.013
g_c	$[0, 1]$	0.1
g_m	$[0.01, 1]$	0.1

Table 7.1: Permitted values and deviations of control genes

As in [49] the mutation has two phases. In the first phase, the mutation of the control genes is performed. Then other genes are mutated according to the new values of the control parameters. The crossover probability control works as follows. Firstly two candidate solutions are selected by a tournament selection. Then the mean of their g_c genes is taken as the probability of crossover P_c for these two solutions

$$P_c = \frac{g_c^1 + g_c^2}{2}. \tag{7.6}$$

7.3.3 Variants of multiobjective evolutionary estimation

In order to obtain the best performance, we propose three variants of the evolutionary estimation of missing values in TDM. The first variant is based on the multi-objective genetic algorithm and doesn't use the self-adaptation. The second variant utilizes the self-adaptation. The first two variants start with a randomly generated initial population. The third variant uses quadratic programming approach to generate the initial population. The result of QP approach is transformed into chromosome and this chromosome is copied into the initial population. The third variant also uses self-adaptation.

7.4 Experimental results

In order to evaluate the proposed methods, two data sources are utilized: (1) field data from annual manual counting from the city of Prague (counting in year 2008 and 2009). The data cover the central part of the city which is modelled using 126 nodes (28 of them

are border nodes) and 277 edges (117 of them without the traffic intensity). This Prague dataset was described in section 6.3. (2) synthetic data where three random scenarios of incomplete TDMs were generated. These syntetic maps have 200, 500 and 1000 nodes.

7.4.1 Performance analysis

For the real scenario from Prague, a solution produced by the QP method has objective values: $F_1 = 5.286681E8$ and $F_2 = 152450$. We used the Octave QP solver to obtain this solution.

Three different variants of multiobjective GA were analyzed: (1) Genetic algorithm starting with a randomly generated initial population without self-adaptation; (2) GA starting with a randomly generated initial population, but self-adaptation is enabled; (3) GA starting with the initial population generated by quadratic programming and self-adaptation is enabled. All three variants optimized two objective functions: (i) the error on nodes as described in Eq. 7.4 and (ii) the sum of differences on edges against last year counting (Eq. 7.5). The population size was 50 for each run.

Gene-ration	Error on nodes F_1	History difference F_2	Mean node error		Mean history difference	
			absolute	relative	absolute	relative
Basic genetic algorithm						
1000	$6.813175.10^9$	165515	4488	30.4 %	1444	28.4 %
2000	$6.006503.10^9$	107727	3746	19.0 %	931	13.0 %
5000	$1.669960.10^9$	39446	2365	14.6 %	338	8.2 %
10000	$7.082570.10^8$	27780	1800	12.5 %	246	6.9 %
25000	$6.081299.10^8$	21955	1672	11.5 %	193	5.8 %
40000	$5.922080.10^8$	19916	1645	11.3 %	167	5.3 %
50000	$5.885092.10^8$	18223	1629	11.3 %	160	5.1 %
Genetic algorithm with self adaptation						
1000	$2.380441.10^9$	28188	2546	14.8 %	279	5.6 %
2000	$9.907829.10^8$	8765	1842	12.1 %	58	1.6 %
5000	$5.894440.10^8$	817	1568	10.3 %	8	0.2 %
10000	$5.319566.10^8$	57	1473	9.7 %	2	0.1 %
25000	$5.289430.10^8$	9	1446	9.5 %	0	0.0 %
40000	$5.286752.10^8$	0	1445	9.5 %	0	0.0 %
50000	$5.286680.10^8$	0	1445	9.5 %	0	0.0 %
Quadratic programing and genetic algorithm with self adaptation						
1000	$5.286681.10^8$	54754	1445	9.5 %	358	11.3 %
2000	$5.286681.10^8$	19044	1445	9.5 %	175	6.5 %
5000	$5.286681.10^8$	5488	1445	9.5 %	41	1.8 %
10000	$5.286681.10^8$	550	1445	9.5 %	11	0.5 %
25000	$5.286681.10^8$	7	1445	9.5 %	1	0.0 %
40000	$5.286681.10^8$	2	1445	9.5 %	0	0.0 %
50000	$5.286681.10^8$	1	1445	9.5 %	0	0.0 %

Table 7.2: Fitness values and solution properties for real world Prague data (mean from 100 runs).

Table 7.2 contains results for Prague dataset. It gives the mean of the best values of objective functions and other parameters after a specified number of generations (1000, 2000, 5000, 10000, 25000, 40000 and 50000) from 100 runs of GA. The maximal allowed difference against history was 30 %. Moreover, we provide information about the mean error on nodes and mean difference against history for edges. The mean absolute node error is the mean difference between the number of incoming vehicles and outgoing vehicles for nodes. The relative node error is obtained by comparing the absolute node error with the number of incoming vehicles for each node. Similarly, we calculate the absolute mean history difference as the mean difference between historic and estimated intensity. The relative historical difference is obtained by comparing the absolute historical difference with the intensity for each edge.

The results for synthetic data are provided in the Table 7.3. The synthetic scenarios were generated completely randomly and thus the results cannot be used for a direct evaluation of the method. However, these results can be useful for comparison of convergence speed. For synthetic data we provide only the values of objective functions.

	Basic genetic algorithm		GA with self adaptation	
Generations	Error on nodes F_1	History difference F_2	Error on nodes F_1	History difference F_2
Random scenario with 200 nodes				
2000	$9.713179.10^9$	26802	$9.705400.10^9$	7929
10000	$9.680005.10^9$	7771	$9.672611.10^9$	43
25000	$9.676344.10^9$	5233	$9.672601.10^9$	1
Random scenario with 500 nodes				
2000	$3.370222.10^{10}$	273080	$3.353729.10^{10}$	86723
10000	$3.339072.10^{10}$	54241	$3.332559.10^{10}$	2427
25000	$3.336195.10^{10}$	33805	$3.332383.10^{10}$	82
Random scenario with 1000 nodes				
2000	$9.354801.10^{10}$	854547	$9.103709.10^{10}$	450027
10000	$8.909628.10^{10}$	244548	$8.855298.10^{10}$	64569
25000	$8.884524.10^{10}$	155489	$8.853087.10^{10}$	15174

Table 7.3: Fitness values for syntetic data (mean from 100 runs).

It can be observed that variant (1) provides the worst results for both objective functions. The variant (2) provides better results against the history, but significantly worse results in node errors than the combined variant (3) approach when 30% constraint is applied. It was possible to find a solution, which has only a 9.5 percent error on nodes. This solution was found by variants (2) and (3). Variants (2) and (3) are also capable to find a solution with a zero difference against the history. The solution with the best difference against history, which was found by a variant (1) has difference $F_2 = 160$.

We visualised the results using box plots in Figure 7.2. The figure shows the best fitness values for 100 runs of GA for different variants. The left part of the figure shows the best values of the errors and the right side the best values for different. It is possible to see, that the variant (1) provides significantly worse results than other variants. One thousand generations takes 5.5 seconds on the Intel Xeon 2.66 GHz processor. It can be seen that GA is slower than QP. A traffic expert solves the same problem in a few days.

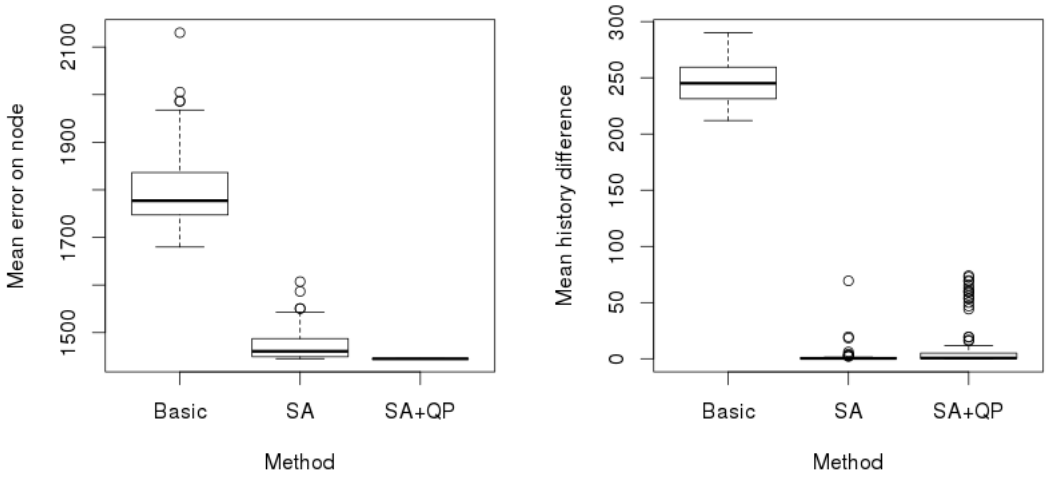

Figure 7.2: Fitness values comparision after 10000 generations.

7.4.2 Pareto front

One goal of the multi-objective optimization is to provide solutions widely spread along the Pareto front at the end of the optimization. Figure 7.3. shows the results obtained from a single run of the genetic algorithm. It shows positions of individual solutions in the whole objective space. On the horizontal axis there is an error on nodes and the vertical axis shows a difference against historic values. Every such point represents one tradeoff solution and it can be seen that solutions are quite well spread.

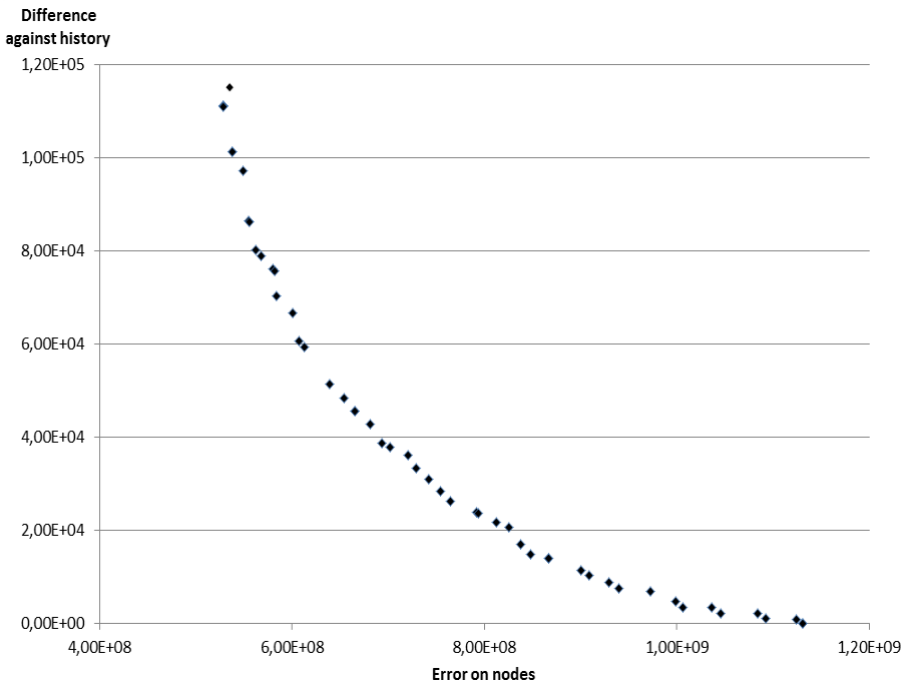

Figure 7.3: Spread of solutions along Pareto front after a single run with 50 solutions in population. Each point represents one solution.

7.5 Discussion

It was shown that two-objective optimization process gives many tradeoff solutions situated on the Parreto front. This is usefull for iterative estimation, because one can choose the best trade-off according to his/her knowledge. Also, it is possible to use the constraints in the same way discussed in the QP approach. The best results were obtained by combining both methods, when the initial solution is generated by the quadratic programming and then further optimized by GA.

Recently built parking lots, shopping centres, etc. can introduce significant errors to TDMs with respect to historical data. The errors on nodes corresponding to these new entities are obviously much less significant than on other nodes. Automatically generated estimation can't deal with this fact. The proposed approach is able to incorporate this kind of domain knowledge into the solution. In one of supported scenarios, a traffic expert is allowed to choose from three methods to define the error on each single node n. These options can be described by Equation 7.7, where $k \in \{1, 2, 3\}$.

$$E_n = \left| \sum_{e \in I_n} d_e - \sum_{e \in O_n} d_e \right|^k \tag{7.7}$$

7.6 Java application for estimation of missing values

We developed a computer program in Java, which uses an incremental process of traffic density estimation. This process is supposed to be driven by a user – traffic expert. At the begining the user sets the values for measured edges and runs the multiobjective genetic optimization process. There are several optimized solutions at the end of this process. One of them can be chosen and eventually edited. The user can also change importance of the errors on nodes and constraints as mentioned previously. After this editing, the optimization process can be performed again and again. This iterative and interactive process continues until a sufficient estimation is reached.

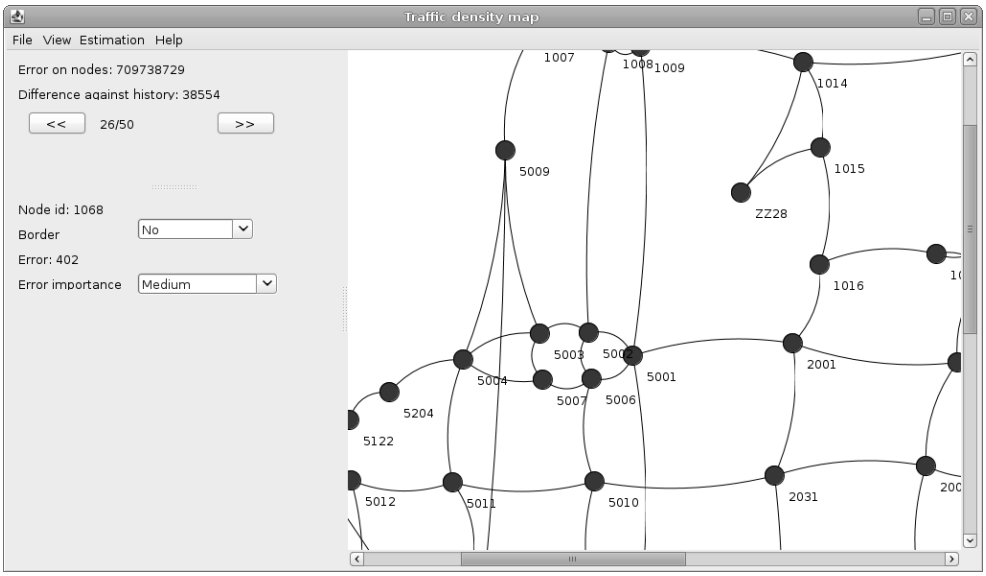

Figure 7.4: A screenshot of application for estimation of missing values in TDM.

Chapter 8

Multiobjective Selection of Input Sensors for SVR Applied to Road Traffic Prediction

8.1 Motivation

Modern traffic sensors can measure various road traffic variables such as the traffic flow and average speed. However, some measurements can lead to incorrect data which cannot further be used in subsequent processing tasks such as traffic prediction or intelligent control. In this thesis, we propose a method selecting a subset of input sensors for a support vector regression (SVR) model which is used for traffic prediction. The method is based on a multimodal and multiobjective NSGA-II algorithm. The multiobjective approach allowed us to find a good trade-off between the prediction error and the number of sensors in real-world situations when many traffic data measurements are unavailable.

8.2 Method

The proposed method can be used to either predict the traffic flow or estimate missing values for a broken sensor. In the first phase, the SVR model is trained using historical data (the train set) in the supervised learning scenario. Trained SVR model then describes mathematical dependencies among the values of the sensor for which predictions are desired and other sensors in the area. Other historical data, unseen during the learning phase (the test set), are used to validate the resulting model. The multiobjective multimodal NSGA-II algorithm is employed to find the proper subset of input sensors for the SVR model.

Traffic data are usually available as a set of time series $s_1, \ldots, s_n$; one time series for each variable measured by a traffic sensor. In order to train the SVR model, it is necessary to convert these data into training samples (Fig. 8.1). By means of a sliding window, the current value ($s_i^{(0)}$) and a few (h) previous values ($s_i^{(-1)}, \ldots, s_i^{(-h)}$) from each series are taken into a training sample. In the case of estimating the current value of a broken sensor (Fig. 8.1, left), the current value $f^{(0)}$ is included into the training sample as a dependent variable. In the case of traffic forecasting in the place of sensor, the future value $f^{(+l)}$ is included into the training sample (Fig. 8.1, right), where l represents the prediction horizon.

We employed the multiobjective multimodal NSGA-II [27] operating over binary strings to select proper input sensors for SVR. Each gene represents one input sensor, where 1

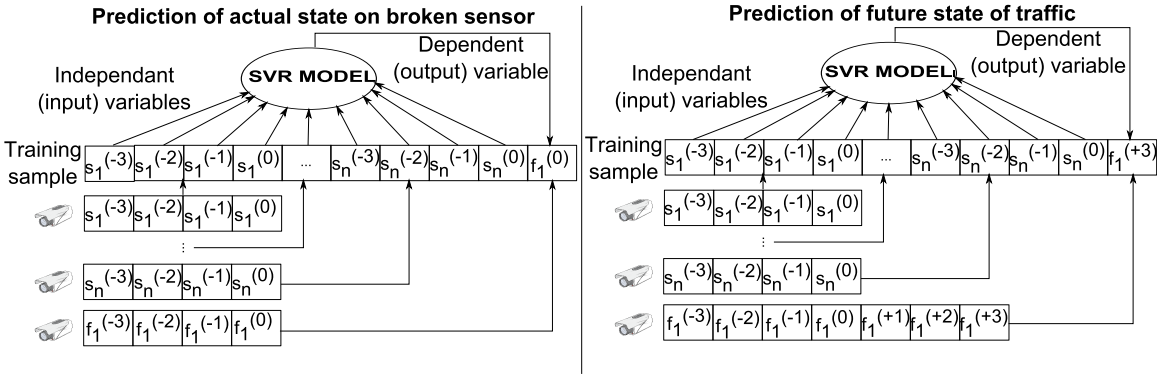

Figure 8.1: Composition of training samples for SVR: prediction of a current value (left) and prediction of a future value (right) of a sensor producing f.

denotes including and 0 excluding of a particular sensor from the input vector fed to SVR (Fig. 8.2).

Three objectives are considered (all to be minimized) – the number of sensors used as inputs for SVR, the rate of missing samples for prediction and the prediction error. The rate of missing samples is portion of time for which the concrete model can't be used because of missing data. Two well-known error metrics can be used as error objective function: root mean squared error (RMSE) or relative squared error (RSE). All objectives are evaluated using the test set.

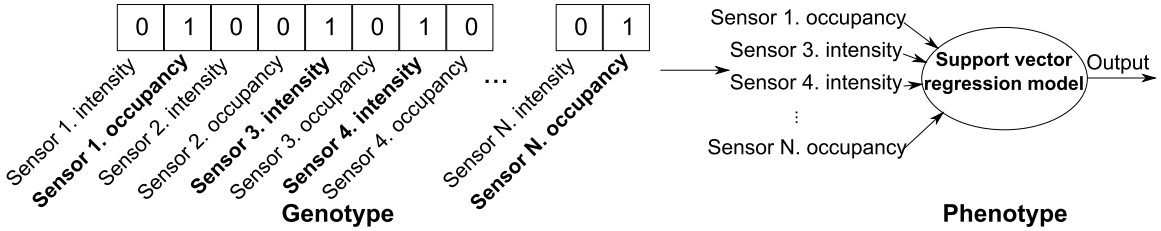

Figure 8.2: Chromosome encoding and a corresponding phenotype (SVR model)

8.3 SVR Parameters Settings

Although the optimization of SVR metaparameters is not the primary objective of this work, we tried to identify the most suitable setting of basic parameters of SVR which employs radial basis kernels (RBF). Figure 8.3 shows RSE for various settings of the regularization parameter ($C = \{2^{-5}, 2^{-4}, \ldots, 2^{14}, 2^{15}\}$) and kernel parameter ($\gamma = \{2^{-15}, 2^{-14}, \ldots, 2^{2}, 2^{3}\}$). In the following experiments we will utilize $C = 2^3$ and $\gamma = 2^{-12}$ because a clear minimum of RSE can be seen for them in Fig. 8.3.

8.4 Method Evaluation

The proposed method was evaluated on places 6, 11, 19, 22, and 23 of the Seattle area (see chapter 6.2). For each sensor located on these places, four SVR models were created. The first two SVR models are trained to perform a short-term prediction in the horizon of

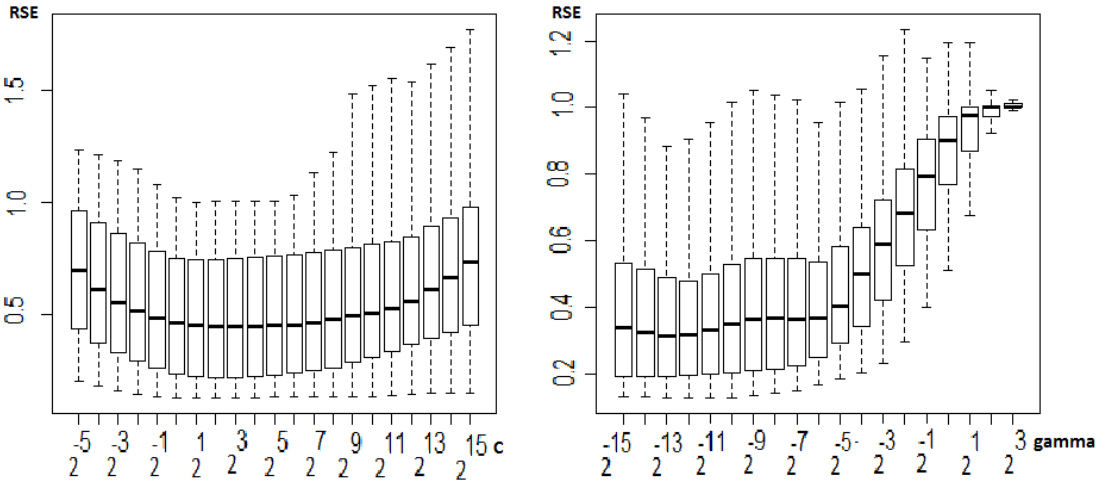

Figure 8.3: The effect of setting of the regularization parameter C and kernel parameter γ on the quality of SVR prediction.

15 minutes. One of them uses only the actual values measured on the neighbor detectors in the area and the second one uses the actual values and the values measured on these sensors in previous 15 minutes. The other two SVR models are trained to estimate the actual value on the sensor in the case of a sensor error. And again, one of them uses only the actual values measured on the neighbor detectors in the area and the second one uses the actual values and the values measured on these sensors in previous 15 minutes.

The parameters of the NSGA-II genetic algorithm are as follows. The probability of uniform crossover is 70% and the probability of mutation is 5%. Each NSGA-II run, which operates with a 40 member population and 100 generations (4000 fitness evaluation), is repeated 20 times. The prediction error is given as the RMSE. The evolution utilizes approximately 50% of the available data to train SVR model, the remaining data are used to validate the evolved SVR models in the following figures and tables. Experiments were

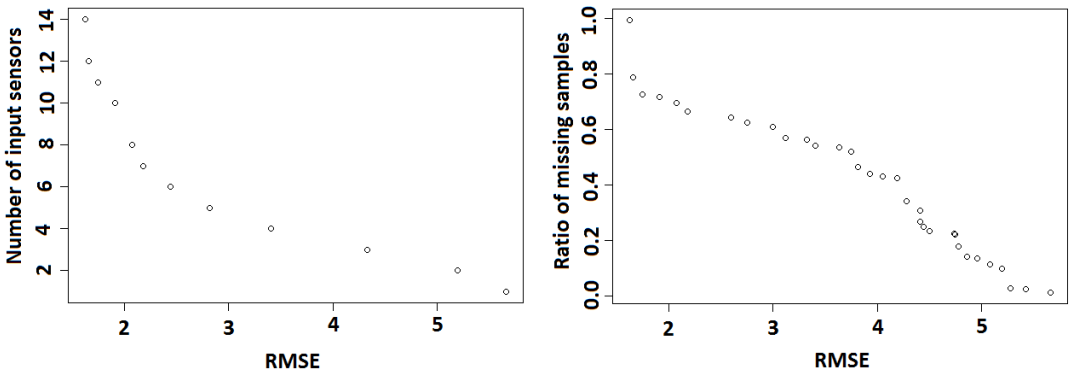

Figure 8.4: Non-dominated compromises obtained for the future traffic forecasting scenario with the prediction horizon of 15 minutes for sensor number 3 measuring the traffic flow on place 19.

performed on an Anselm supercomputer whose nodes are equipped with two Intel Sandy Bridge E5-2665 chips. These chips contain 8-core processors working at 2.4 GHz. One run takes approximately 10 hours of one processor core. Our software was implemented in the scripting language of the system R for statistical computing [106]. We used publicly available R package e1071 for training of SVR models.

Figure 8.4 shows the resulting Pareto fronts from a typical NSGA-II run. Numerous non-dominated compromises between RMSE and the number of input sensors (left) and RMSE and the ratio of missing samples (right) are shown. The results were obtained for the future traffic forecasting scenario with the prediction horizon of 15 minutes for sensor number 3 measuring the traffic flow on place 19. The predicted values and correct values for one example solution are shown in Fig. 8.5.

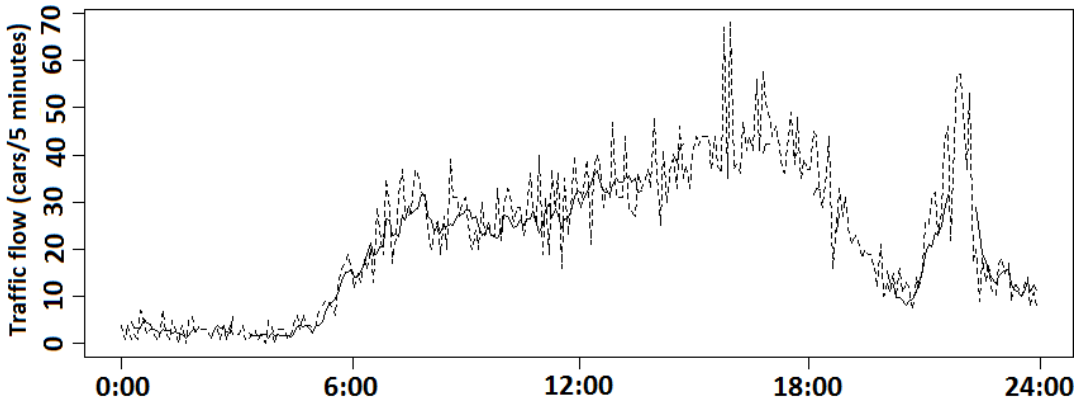

Figure 8.5: Predicted values (dashed line) and correct values (normal line) of traffic flow for sensor 3 on place 19 on July 1^{st}, 2011.

Experiments show that the proposed method, in contrast with a common approach reported in the literature, can provide reasonable results even if many samples are unavailable. The best results obtained from 20 independent runs of NSGA-II are presented as box plots in Fig. 8.6. Resulting RMSE values are shown for the traffic flow and occupancy ($l = 3$, $h = 3$) when less than 10%, 30%, 50%, and 70% samples are unavailable. The results are given for sensor 3 on place 11. It is important to note that, for example, the value of 70% means that for a given SVR model the samples from the test set are incomplete in $24 \cdot 0.7 = 16.8$ hours of a day, i.e. the SVR model will not work for most of the time.

In order to provide results for some other sensors, Fig. 8.7 summarizes the best RMSE values obtained for places 6, 11, 19, 22, 23. For each place, 3 sensors exhibiting the biggest mean traffic flow and occupancy were chosen. It can be seen that RMSE increases when more samples are available in the test set. The short term traffic prediction scenario with horizon of 15 minutes and 15 minute history is considered in the figure.

And finally, Fig. 8.8 summarizes the mean RMSE over all sensors on all prediction places in all considered scenarios. The columns are:

- *actual* – the prediction of the actual values on a broken sensor using the actual values on sensors from other places ($l = 0$, $h = 0$)
- *actual (15 min.)* – see actual, but in addition, some historical data are used ($l = 0$, $h = 3$)

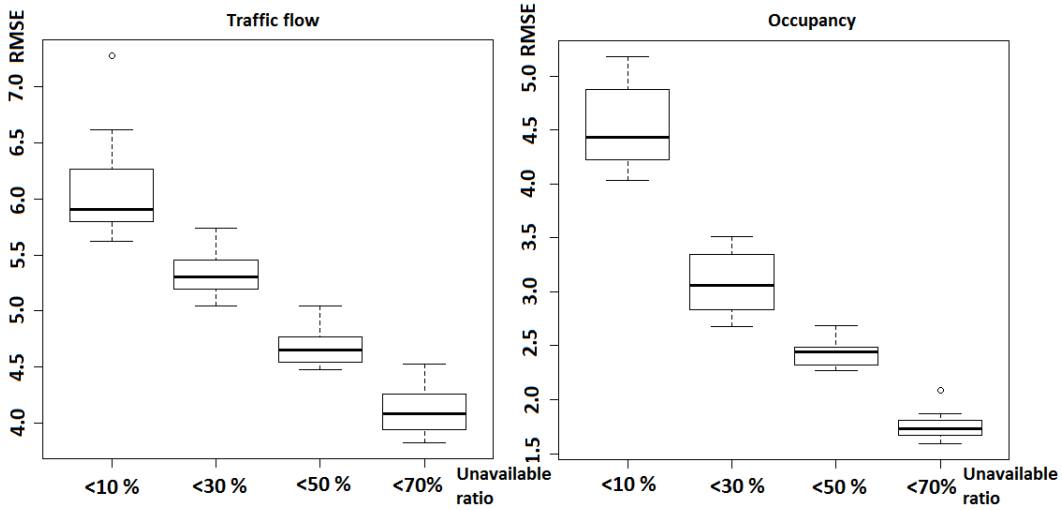

Figure 8.6: Prediction error (RMSE) when less than 10%, 30%, 50%, and 70% samples are unavailable from sensor 3 on place 11.

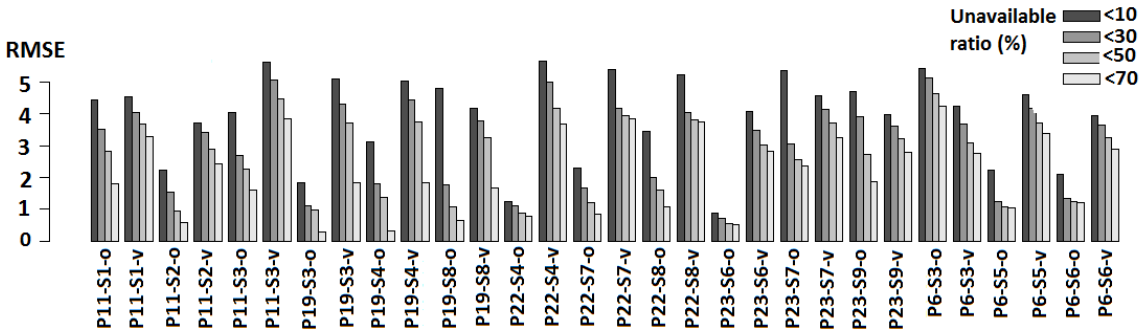

Figure 8.7: Prediction error (RMSE) at 5 places (each with 3 sensors) for different amount of unavailable samples (v – traffic flow, o – occupancy).

- *15 min. future* – the traffic prediction in near future with a 15 minute prediction horizon ($l = 3$, $h = 0$). As the input for SVR model the actual values on other sensors were used.
- *15 min. future, 15 min. history* – see the previous one, but the historical data are used ($l = 3$, $h = 3$).

8.5 Comparison with a Single Objective GA

In order to justify the multiobjective approach, we consider a single criterion optimization scenario, in which RMSE is used as the only fitness function. The single-objective GA works with 40 individuals in the population, the probability of crossover is 70%, the probability of mutation is 5%, and 2-individual tournament selection (with elitism) is chosen. Table 8.1 compares NSGA-II with the single objective GA for several places and sensors (the best values from 20 independent runs are reported). It can be seen that the single objective GA tends to provide solutions with very small RMSE values; however, it opportunistically

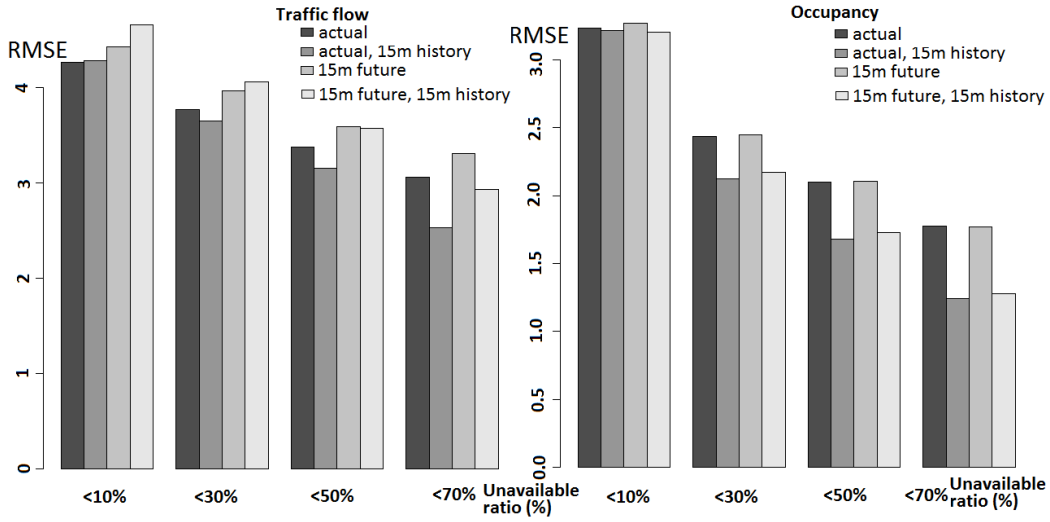

Figure 8.8: Mean RMSE over all sensors on all prediction places in all considered scenarios

Location			Multiobjective approach RMSE for Unavailable ratio:				Best single objective GA result	
Place	Sensor	Variable	$< 10\%$	$< 30\%$	$< 50\%$	$< 70\%$	RMSE	Unavailable ratio
Current values on sensor.								
11	3	traffic flow	5.27	4.63	4.16	4.01	2.66	96.9
11	3	occupancy	3.81	3.50	3.31	3.31	0.31	99.4
22	4	traffic flow	5.33	4.86	4.31	4.20	1.48	99.4
Prediction horizon 15 min.								
11	3	traffic flow	5.50	4.9	4.37	4.23	2.96	97.2
11	3	occupancy	4.02	3.57	3.41	3.35	0.33	99.4
22	4	traffic flow	5.51	4.89	4.56	4.35	1.84	99.0
Current values on sensor, 15 min. history.								
11	3	traffic flow	5.20	4.58	3.91	3.34	1.15	99.4
11	3	occupancy	4.04	2.72	2.18	1.50	0.19	99.4
22	4	traffic flow	5.62	4.71	4.09	3.37	1.04	99.4
Prediction horizon 15 min., 15 min. history								
11	3	traffic flow	5.62	5.05	4.48	3.82	1.17	99.4
11	3	occupancy	4.03	2.68	2.27	1.59	0.24	99.4
22	4	traffic flow	5.64	4.98	4.17	3.66	1.15	99.4

Table 8.1: The best RMSE on selected sensors and places for NSGA-II (less than 10%, 30%, 50% and 70% samples unavailable) and a single objective GA

exploits the test data containing over 85% missing values (in many cases, over 99%, see the Unavailable ratio column). Such a SVR model will thus be useless in practice, because it will not provide any prediction most of the time. Therefore, the single optimization scenario fails in this task.

Chapter 9

Multiobjective Selection of Input Sensors for Travel Times Forecasting Using Support Vector Regression

Travel time information can be used for various purposes, for example, as a source for different transportation analyses or for drivers who are planning their itinerary. Especialy in the situation when there are two or more possible ways to travel and the driver wants to choose one which will take shorter time.

In this thesis, we propose a new method for *travel time prediction* which uses a *support vector regression*. The inputs of our method are data from license plate detection systems and traffic sensors such as induction loops or radars placed in the area. These traffic sensors enable us to measure traffic variables such as traffic volume, road occupancy, average speed and many others [67]. This method is mainly designed to be capable of dealing with missing values in the traffic data. The method is able to create many different SVR models with different input variables. These models are then dynamically switched according to which traffic variables are currently available. Hence the proposed method provides much reliable predictions than traditional license plate detection methods or regression methods, which use the data obtained by a static set of traffic sensors.

9.1 Methods for travel times prediction

In the past, many methods for travel time estimation have been proposed. In this thesis, we will utilize a license plate based approach and a regression based approach.

9.1.1 License plate based approach

The basic principle of this approach is depicted in Figure 9.1. It is necessary to have at least two cameras. The first camera (A) is placed at the begining of the road segment and the second one (B) is placed at the end of the road segment. Each camera is connected to a pattern recognition software, which is able to read numbers on license plates. The license plate number for each vehicle is stored in a database. When the same license plate number

is detected by both cameras for a given vehicle then the travel time can be calculated [124, 63].

Based on this data, the information of the expected travel time can be displayed to drivers at the begining of the road segment. The main advantage of this method is that it is a quite straightforward principle. The biggest drawback is that the information displayed to drivers is delayed because it is based on conditions which existed in the current place when the last vehicle leaving the road segment was registered by camera A. Another drawback is that the license plate reading systems are not highly reliable and many vehicles pass the segment undetected.

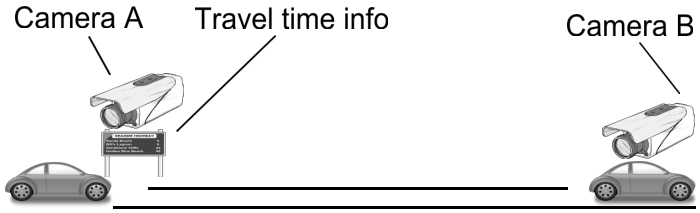

Figure 9.1: The principle of travel times estimation using license plate recognition.

9.1.2 Regression based approach

Another approach to predict travel times is based on the data from traffic sensors such as radars, loop detectors, etc. Based on historic experience, the regression approaches predict the expected value of travel times using traditional supervised learning techniques known from the machine learning area. For example, the regression model can be constituted using neural network or support vector regression [143, 29].

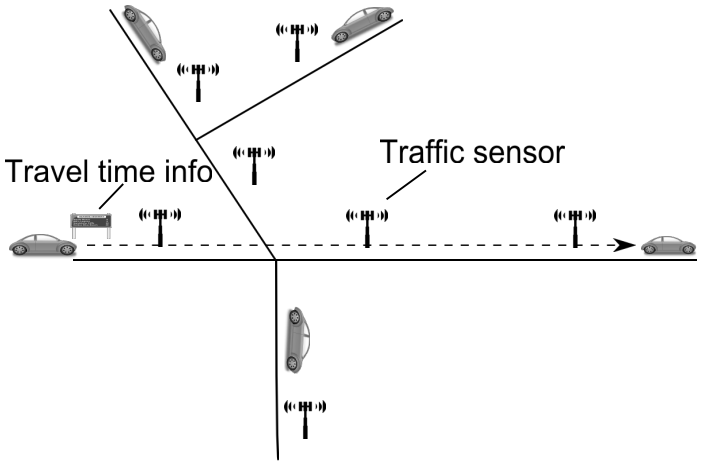

Figure 9.2: The principle of travel times estimation using traffic sensors in the area.

9.1.3 Problem of missing values in traffic data

The quality and availability of both methods described for travel times estimation is largely dependent on the quality of the input data. In the license plate based approach, the quality of prediction is largely dependent on the quality of the pattern recognition software for

license plate reading. Sometimes the license plate is not read and the travel time of this car can't be measured. Another problem appears when the car stops inside the measured section for example, in a shopping center etc. In this case a very long travel time is assigned to the car. We call these measured travel times as outliers and it is necessary to filter these values out. These two problems can lead to missing predictions if the number of unavailable vehicle travel times is high.

The data produced by sensors such as loop detectors, radars and camera detectors are not absolutely reliable and there are many missing values. Unfortunately, support vector regression and many other regression methods can not deal with the missing values. If one input variable is missing the support vector regression algorithm does not produce any result. In our approach, we will focus on selection of proper input sensors for travel times prediction. The proposed method is multiobjective and one of the objectives is to minimize the amount of missing samples in the input data for prediction.

9.2 Search for subsets of input variables

In order to predict travel times, SVR with a radial basis kernel is exploited. Potential input variables for SVR are the data measured by traffic detectors in the area and travel times estimated by a license plate recognition system. The main goal of the optimization process is to minimize the travel time prediction error. However, as it was mentioned earlier, the data produced by license plate recognition systems or traffic detectors are sometimes unavailable, so it means that if any input value for this model is missing the model will not work. Some of these systems are more reliable than others. Because of it we also decided to minimize the time for which the prediction of the SVR is not available and the number of input variables. The goal is to find various trade-offs. Some of them will provide more accurate prediction, but will require that more input variables are available. On the other hand, there will be less accurate solutions which will use fewer input variables and because of it their prediction will be available for a longer period of time.

The basic principle of our method is shown in Figure 9.3. The traffic data measured by traffic sensors (volume, occupancy and average speed) are ussualy aggregated into time intervals, for example, 5 minute intervals. We denote time series provided by traffic sensors as $s_1 \dots s_n$. These series consist of measured values. The actual value of serie s_i is denoted $s_i^{(0)}$, the value measured in the previous time interval is denoted $s_i^{(-1)}$ etc. The license plate system also provides series of data, which will be denoted LP. The value $LP^{(0)}$ is the average travel time of vehicles, which were detected by the second camera (B) in the current time interval. The inputs (independent variables) of SVR are the correct values and a h previous values. The output value (dependent value) is the value of the travel time. It should be noted that the correct value of the travel time can be reconstructed using historical data during the model construction.

However, not all input sensors are useful for prediction and there can be sensors which are often broken and don't provide any data. Hence, it is necessary to choose a subset of available sensors which will serve as inputs for the SVR model. The multimodal NSGAII algorithm will perform this task. In order to utilize genetic algorithms, we here define a suitable representation of candidate solutions. In our approach, we represent candidate solutions as binary strings. One bit corresponds to one input variable which is either a sensor or license plate detection system. If the bit is set to one, then the corresponding variable is used in the SVR model. Otherwise, the variable is not used as the input for the SVR model. An example of a binary string representing one candidate solution is depicted

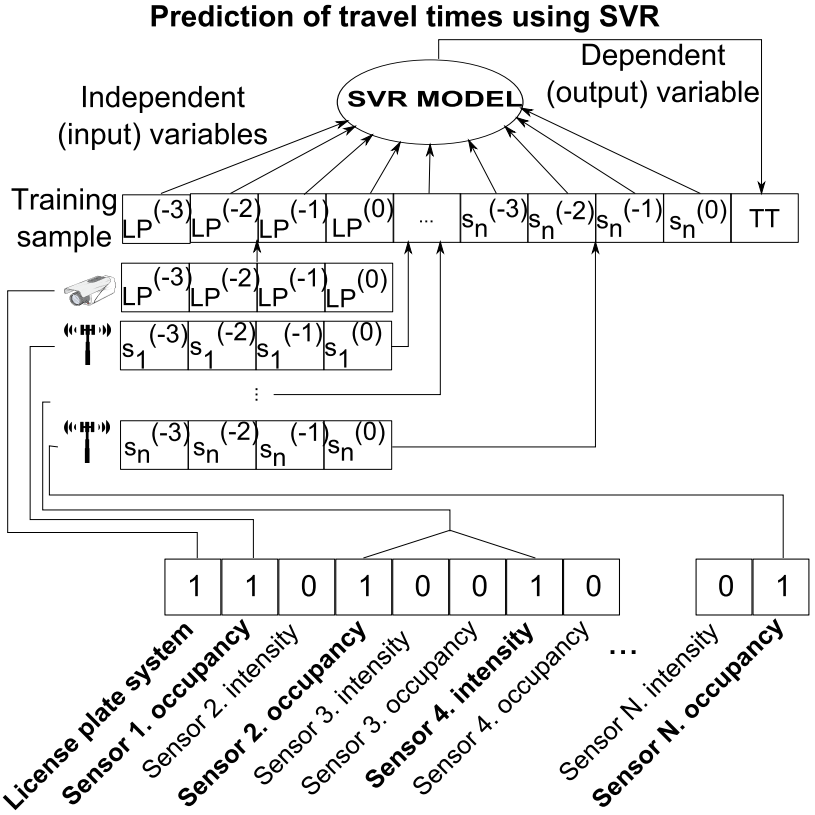

Figure 9.3: Problem encoding for prediction of travel times using SVR.

in Figure 9.3. The selected input variables for SVR are shown in bold.

The proposed multimodal NSGAII algorithm utilizes the uniform crossover and bit flip mutations. The goal is to minimize three objective functions: the quality of prediction, the unavailability ratio and the number of input variables. The quality of prediction is usually measured by either the root mean square error (RMSE) or relative squared error (RSE). The unavailability ratio is the portion of time, for which the trained SVR cannot be used, because at least one of its input variables is missing.

We will utilize the traditional supervised learning scenario [91] in which the data are divided into two sets. The first set is called the training set and it is used to calibrate the SVR model (training in terms of machine learning methods). The second part of the data is used for validation of the quality of prediction. In our approach, we evaluate the objective functions on a test data set. Figure 9.4 shows the process of evaluation of the fitness functions in one generation of the GA.

9.3 Dynamic model switching

The reason why it is good to have multiple models with different characteristics is that we can switch among them during the real time prediction process according to a given situation. The main factor which is changing over the time is data availability from sensors. The key idea of our approach is to dynamicaly switch among these models according to which data are currently available. The highest-quality model is activated if possible. If the input data for this model are missing, the second best model is taken. If the input data

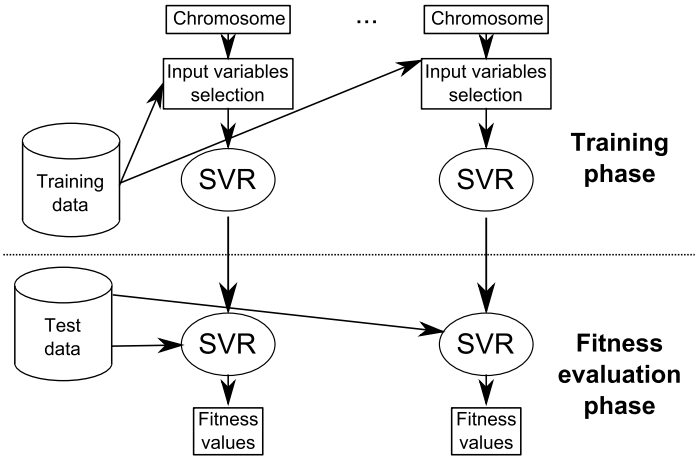

Figure 9.4: Evaluation of fitness functions in one generation of the GA.

for this model are not available, we will use the next model. We can continue with this process, until we get the model for which the data are available, or we have to stop because the data are unavailable. We can also define a syntetic treshold to define a minimal quality of the SVR model. If the quality of the model is worse than this treshold, the result is that the prediction of travel times is not available. The principle of model switching is shown in Figure 9.5.

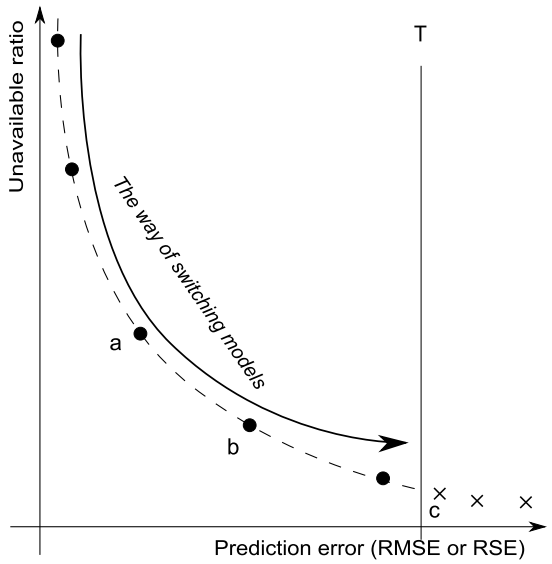

Figure 9.5: The principle of model switching.

9.4 Experimental results

9.4.1 Data preprocessing

The proposed method was evaluated using real world data from Seattle (see Chapter 6.2). We will perform our experiments on two trajectories denoted as A and B. These trajectorise

were described in Chapter 6.2.5. Before the evaluation of the proposed method is conducted, the raw data has to be preprocessed. For sensors data, we removed all records showing a non-zero flag from the dataset and aggregated measured values from one minute intervals to five minute intervals. As the aggregation functions we applied the sum of the traffic volume and the mean of occupancy and speed. In the case of license plate recognition data, we at first phase filtered out the outlier values. These outliers appear mainly because drivers can stop somewhere inside the area for some reason. The percentil characteristic of the travel times in section A is depicted in Figure 9.6. We filtered out the values higher than 98 percentil. The rest of the data was aggregated to 5 minute intervals using the mean aggregation function.

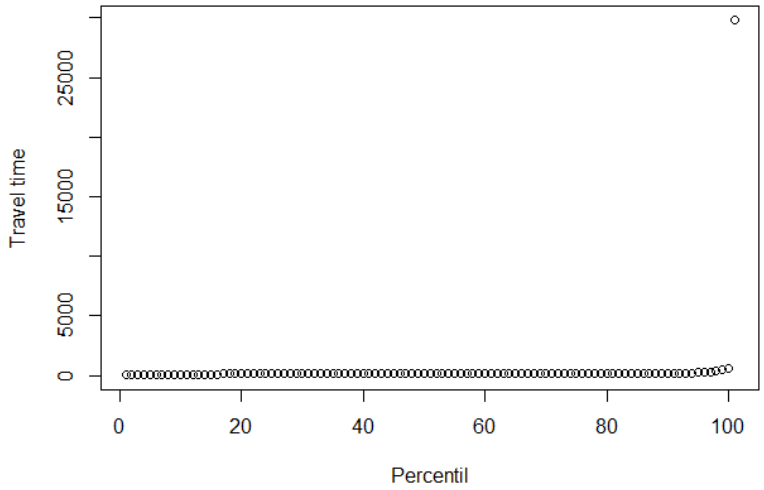

Figure 9.6: Percentil values of travel times measured in section A.

9.4.2 Search for the set of input variables

To evaluate the proposed method, we firstly implemented the method in a scripting language of a system for statistical computation R [106]. Experimental evaluation consisted of several steps. First we were searching for the best parameters of GA and SVR. Most parameters were set according to our previous experiments (see Chapter 8.3). They are summarized in Table 9.1. Table 9.2 gives the obtained prediction error (RMSE) and availability for various probabilities of crossover and mutation.

Then we evaluated the ability of the multiobjective genetic algorithm to find different sets of input variables for the SVR applied to travel times prediction. For road segments A and B we tested four variants of our algorithm, which differ in the error metrics (RMSE vs. RSE) and the number of history samples h of input variables (described in Figure 9.3). 80 independent runs were performed for each experiment. 100 iterations of the genetic algorithm were generated. Experiments were carried out on the Anselm supercomputer whose nodes are equiped with two Intel Sandy Bridge E5-2665 procesors. The 100 iterations of our algorithm take approximately 12 hours on a single core.

Figure 9.7 (left) shows a set of trade-off solutions obtained at the end of one run of the

Genetic algorithm	
Iterations	100
Population size	40
Type of crossover	uniform
Probability of crossover	70%
Probability of mutation	bit flip
Type of mutation	5%
Support vector regression	
Kernel type	radial basis
Reguralisation parameter C	2^3
Kernel parameter γ	2^{-12}

Table 9.1: Parameters of genetic algorithm and support vector regression.

Crossover prob.	**Mutation prob.**	**Availability**	**RMSE**
0.3	0.01	0.81	26.77
0.3	0.05	0.63	26.87
0.3	0.10	0.70	27.24
0.3	0.20	0.68	27.06
0.5	0.01	0.95	26.90
0.5	0.05	0.82	26.61
0.5	0.10	0.78	26.72
0.5	0.20	0.71	27.22
0.7	**0.01**	**0.96**	**26.73**
0.7	0.05	0.83	26.39
0.7	0.10	0.88	26.60
0.7	0.20	0.64	27.43
0.9	0.01	0.92	26.65
0.9	0.05	0.95	26.62
0.9	0.10	0.91	26.67
0.9	0.20	0.91	26.70

Table 9.2: Mean RMSE and availability for various probabilities of crossover and mutation.

multiobjective genetic algorithm, which uses RMSE, the number of input variables and the rate of missing samples as objective functions (only the trade-offs between RMSE and the rate of missing samples is shown). Figure 9.7 (right) shows the trade-offs for an approach which uses RSE and the rate of missing samples. One can observe that some solutions show low error, but they are unavailable for most of the time. On the other hand, there are solutions with higher error, but these solutions can be used for most of the time.

9.4.3 Dynamic model switching

Figure. 9.8 (top) shows the predictions of travel times for 2 days at segment A. The dashed line shows the real travel time reconstructed from the historical data. The solid line shows the travel time predicted by our method. Figure 9.8 (bottom) shows RMSE for currently used model, when the dynamic model switching is activated. It is possible to see that the model is switched quite often.

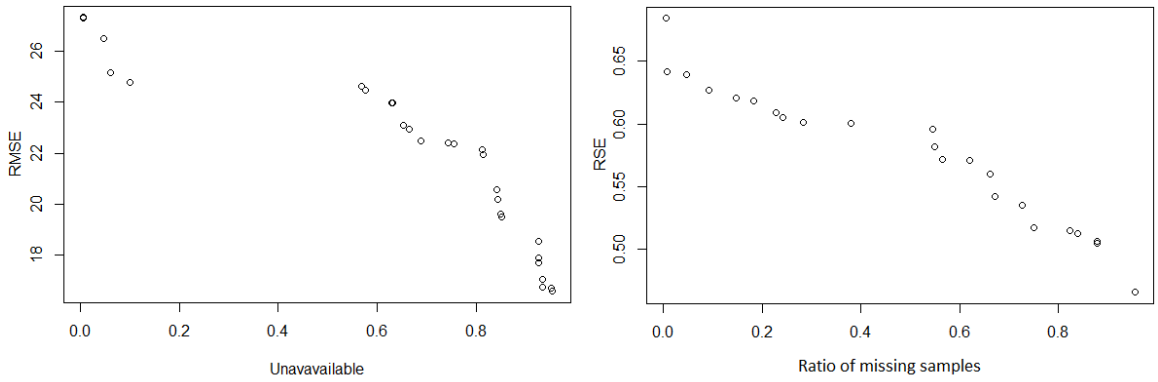

Figure 9.7: Left - trade-offs between RMSE and the rate of missing samples. Right - trade-offs between RSE and the rate of missing samples. Results are obtained from one run of the GA.

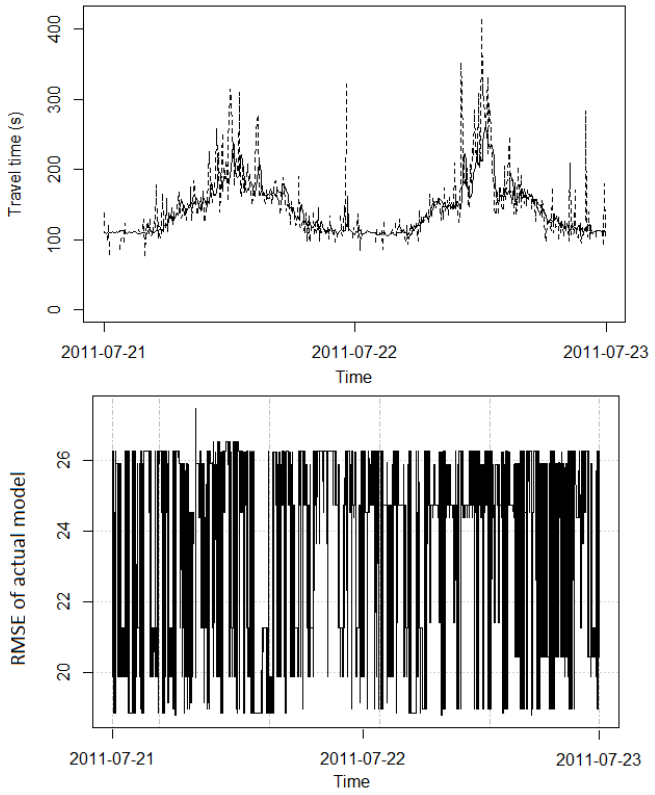

Figure 9.8: True and predicted travel times (top) and model switching (bottom).

Finally, we evaluated and compared different variants of our method for travel times prediction. The aim was to decide if it is better to use RMSE or RSE and if the values from the near history can improve the prediction. Hence, we perform 80 independent runs for each setting which allows us to statisticaly compare the variants. Figure 9.9 shows boxplots for each variant. Boxplots on the top of Figure 9.9 give RMSE obtained by models employing model switching. Each run of the GA provided one value. Boxplots on the bottom of Figure 9.9 show the unavailable ratio. On the basis of these results, it can be

seen that the best results are provided by the variant which uses RSE as the error metric and three values of historical data.

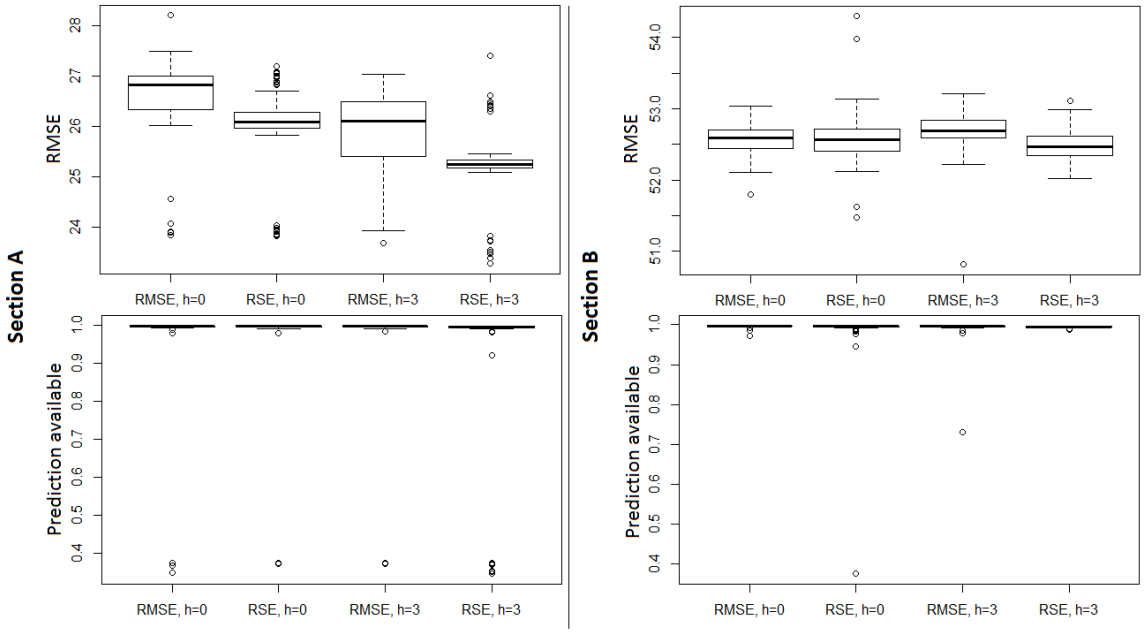

Figure 9.9: Statistical evaluation of RMSE and availability for sections A and B.

9.4.4 Comparison to the license plate approach

We compared our new method to a simple license plate recognition method that we implemented according to literature [63]. The results in terms of availability and RMSE are summarized in Table 9.3, which gives the median values from 80 runs for each experiment.

Method	**Available**	**RMSE**
Section A		
License plate method	0.65	33.22
Our method (median)	0.99	25.23
Section B		
License plate method	0.50	63.81
Our method (median)	0.99	52.47

Table 9.3: Comparission of our algorithm with a simple license plate approach (median values).

9.5 Summary

Proposed method was capable to provide results almost all the time for both sections A and B. However, the system based only on license plate reading system was not capable to provide estimation for nearly half of the time. The RMSE provided by proposed method was also better than RMSE provided by license plate approach for both sections.

Chapter 10

Optimization of Meta-parameters of SVR Applied to Road Traffic Forecasting

In Chapter 9, we have shown that it is possible to use a multi-objective optimization to find many SVR models for prediction of traffic variables. These SVR models differ in the set of input variables that they utilize. We optimized only the set of input variables, but the meta-parameters of SVR remained unchanged. However, this is far from the optimum, because for each SVR the optimal settings of the meta-parameters is different. In this chapter, we try to improve the prediction results by simultaneous search for the optimal data inputs for SVR and the optimal meta-parameters in one run of the multiobjective-genetic algorithm. Moreover, we will propose a parallel implementation of this method capable to significantly accelerate the optimization process.

10.1 Method

In order to obtain high quality predictions, it is necessary to properly select the kernel function and set various parameters, where regularization coefficient C is the most important one. If this coefficient were set too strictly, the problem called *overfitting* could occur. The *overfitting* appears when the training algorithm tries to find a very complex model, which is able to predict almost all values from a training set very precisely. However, this complex model does not usually generalize well and have a quite high error for new data. The opposite problem, called *underfitting*, can appear if the regularization coefficient is set too freely. In the case of underfitting, the model is too simple to be able to predict desired variable.

Two types of kernels are supported in our work: linear SVR and SVR with radial basis kernel. For linear SVR it is necessary to optimize only the regularization coefficient C. The radial kernel requires, in addition to C, to optimize the kernel parameter γ.

The method is based on a multimodal and multiobjective NSGA-II algorithm. The whole chromosome is divided into two parts. The first part contains the information about used input variables. Each gene represents one potential input variable as introduced in Chapter 8. The binary part of the chromosome also contains one additional bit, which defines the type of kernel (0 – the linear kernel; 1 – the radial kernel). We use the uniform crossover and bit flip mutations to modify this part of the chromosome. The second part of

the chromosome consists of real values, which are devoted to SVR meta-parameters. The first real value defines the regularization coefficient in the case that linear SVR is used. In this case the value of regularisation coefficient is equal to $2^{C_{linear}}$.

The second parameter is the value of regularization coefficient in the case that radial kernel is used. In this case the regularisation coefficient is equal to $2^{C_{radial}}$. Finally, the third real value is for gamma parameter ($\gamma = 2^{gamma}$). We use SBX crossover and normally distributed mutations to modify the real valued part of the chromosome. The schema of the whole chromosome is depicted in Figure 10.1.

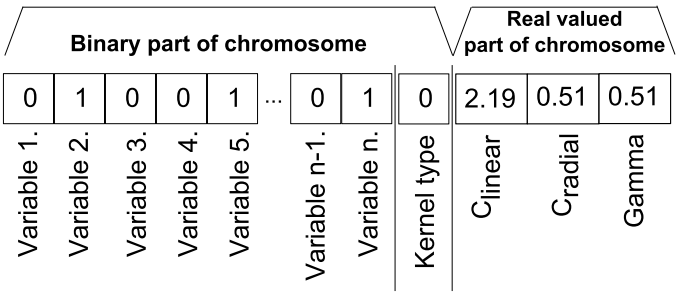

Figure 10.1: Chromosome scheme.

We simultaneously optimize three fitness functions: the quality of prediction (RMSE), the unavailability ratio and the number of input variables. The method provides various compromise solutions at the end of the GA run. Then we utilize the model switching, which was proposed in Chapter 9.3 to obtain prediction results.

10.2 Experimental evaluation

The method was evaluated on three different tasks: data imputation, short term traffic prediction, and travel time prediction. We utilized the model switching described in Chapter 9.3 and Seattle data for the evaluation. The GA parameters are summarised in the Table 10.1.

Parameter	**Value**
Generations	100
Population size	40
Probability of binary crossover	70 %
Probability of binary mutation	1 %
SBX - parameter η_c	2
Real valued mutation - parameter δ	1
Number of runs for each experiment	25

Table 10.1: Parameters of multi-objective GA

10.2.1 Data imputation

The data imputation is a process, in which we try to estimate missing values in data. In our case, the goal is to estimate the missing value for a sensor. The task is simmilar to data imputation task discused in Chapter 8.

We used the Seattle sensys dataset from July 2011 (see Chapter 6.2) for our experiments. In order to simplify the estimation, we aggregated available data into five minute intervals. The first 15 days are considered as training set and the rest is considered as test set. The values of traffic volume are given in the number of vehicles per 5 minutes and occupancy is the portion of time for which the given place on the road is occupied by vehicles. The method was evaluated on the imputation of traffic volume and occupancy data on four intersections (11, 19, 22 and 23). For each intersection, we selected four sensors with the biggest mean value. Table 10.2 shows a comparison with the method which optimizes only the inputs of SVR.

Place Num.	Sensor Num.	Variable Type	RMSE Only Sensors	RMSE Sensors and Meta-param.	Improvement (%)
11	1	occupancy	5.81	5.42	6.73
11	17	occupancy	4.96	4.80	3.14
11	3	occupancy	5.28	4.53	14.06
11	4	occupancy	11.95	11.92	0.28
19	3	occupancy	3.09	2.90	6.20
19	4	occupancy	4.42	4.45	−0.54
19	8	occupancy	8.95	8.70	2.76
22	1	occupancy	3.86	3.69	4.59
22	3	occupancy	4.18	3.94	5.57
22	6	occupancy	2.49	2.49	0.00
22	8	occupancy	3.18	3.00	5.84
23	1	occupancy	11.92	11.72	1.68
23	2	occupancy	8.67	8.40	3.08
23	3	occupancy	10.41	10.29	1.09
23	7	occupancy	8.46	8.40	0.68
11	1	volume	4.94	4.90	0.67
11	13	volume	4.31	4.29	0.48
11	17	volume	3.74	3.65	2.45
11	3	volume	5.62	5.60	0.38
19	3	volume	5.80	5.98	−3.09
19	4	volume	6.42	6.42	0.00
19	7	volume	5.74	5.49	4.33
19	8	volume	4.77	4.78	−0.30
22	1	volume	3.92	3.84	1.99
22	4	volume	5.16	5.32	−3.17
22	7	volume	4.71	4.64	1.55
22	8	volume	4.38	4.32	1.33
23	6	volume	4.11	4.11	0.00
23	7	volume	6.36	6.19	2.66
23	8	volume	6.40	6.30	1.56
23	9	volume	4.53	4.31	5.04

Table 10.2: RMSE obtained in the data imputation task if the selection of input sensors is optimized only and together with SVR metaparameters.

The volume is measured as the number of vehicles per 5 minutes and the occupancy as a portion of time in which the current place is occupied by vehicle. The RMSE corresponds to these variables. The improvement obtained by optimizing SVR meta-parameters is shown in the last column.

The improvement is also depicted in the form of bar plot in Figure 10.2. In which figure the height of each column represents the improvement against method, which optimizes only the set of SVR inputs. The mean improvement for the data imputation is 2.29 %.

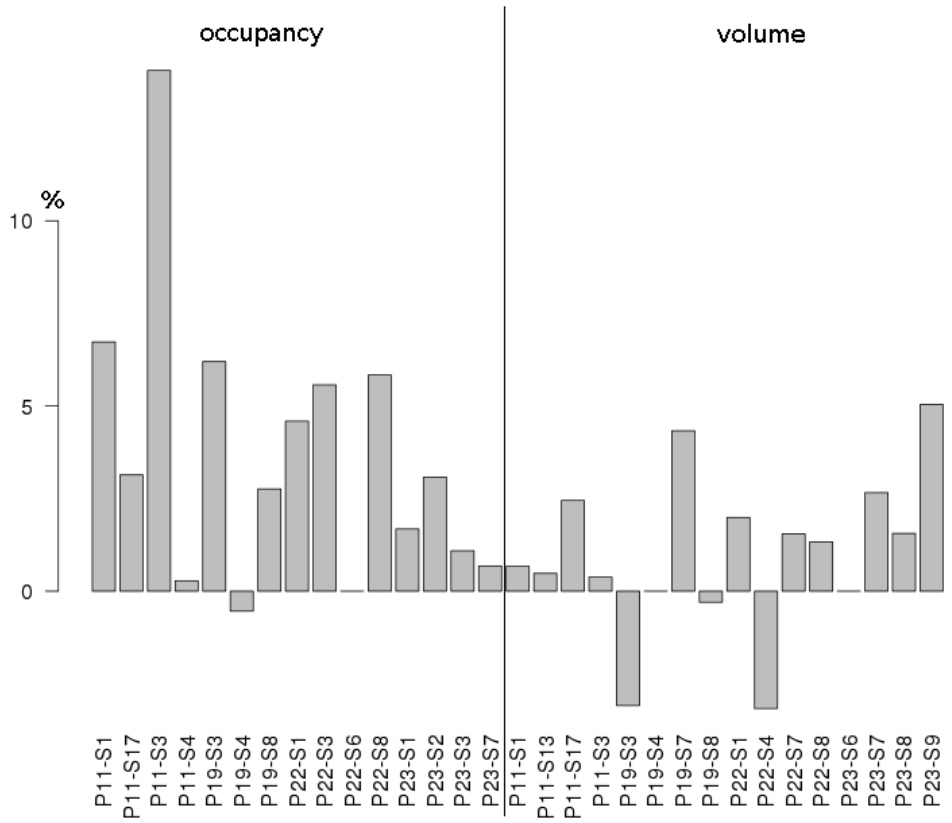

Figure 10.2: Improvement achieved by using simultaneous optimization of SVR inputs and meta-parameters.

10.2.2 Short Term Forecasting of Traffic Variables

The second task to evaluate was traffic forecasting of occupancy and volume with prediction horizon 15 minutes. This task was similar to traffic forecasting task discused in Chapter 8. We used the Seattle sensys dataset from July 2011 aggregated to 5 minutes intrvals for our experiments. Traffic volume and occupancy was forecasted on four intersections (11, 19, 22 and 23). We used four sensors with the biggest mean value of measured variable for each intersection.

Table 10.3 summarizes the results for short time traffic prediction. Mean values calculated from 25 independent runs of NSGAII are reported. The format of the table is the same as for the data imputation. We predicted the values of these variables with the prediction horizon of 15 minutes. We depicted the improvement in the form of bar plot in

Figure 10.3. The mean improvement against the method which optimizes only the set of inputs, is 26.35 %.

Place Num.	Sensor Num.	Variable Type	RMSE Only Feature	RMSE Meta-param.	Improvement (%)
11	1	occupancy	4.18	3.62	13.40
11	17	occupancy	3.65	3.20	12.34
11	3	occupancy	2.47	1.68	32.06
11	4	occupancy	9.62	9.02	6.31
19	3	occupancy	1.59	0.98	38.34
19	4	occupancy	2.49	1.62	34.93
19	8	occupancy	5.84	4.00	31.48
22	1	occupancy	2.42	1.99	17.42
22	3	occupancy	2.38	1.81	23.94
22	6	occupancy	1.13	0.53	53.14
22	8	occupancy	1.87	1.24	33.86
23	1	occupancy	7.11	6.04	15.04
23	2	occupancy	6.50	5.82	10.46
23	3	occupancy	8.02	6.24	22.11
23	7	occupancy	4.54	3.91	13.89
11	1	volume	1.51	1.24	17.97
11	13	volume	2.12	1.46	31.38
11	17	volume	1.98	1.26	36.39
11	3	volume	1.75	1.48	15.44
19	3	volume	3.19	2.15	32.41
19	4	volume	3.21	1.89	41.17
19	7	volume	3.15	2.27	28.06
19	8	volume	2.97	1.73	41.79
22	1	volume	1.84	1.36	25.90
22	4	volume	2.16	1.30	39.84
22	7	volume	2.29	1.56	31.84
22	8	volume	1.91	1.27	35.85
23	6	volume	1.80	1.39	22.58
23	7	volume	2.54	2.14	15.73
23	8	volume	2.67	2.17	18.85
23	9	volume	1.59	1.23	23.04

Table 10.3: RMSE for short term traffic forecasting with prediction horizon 15 minutes.

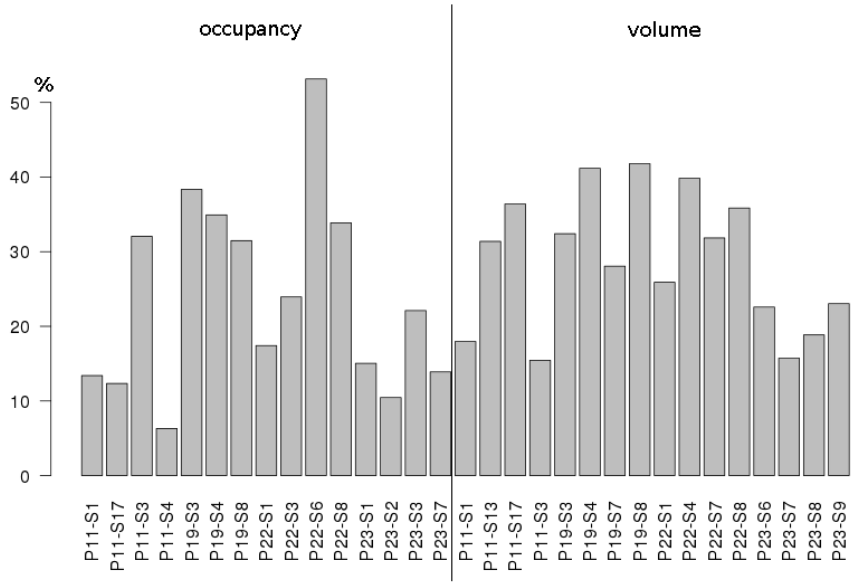

Figure 10.3: Improvement in RMSE achieved by using simultaneous optimization of SVR inputs and meta-parameters. The results for short term traffic forecasting are depicted.

10.2.3 Travel times prediction

The last evaluated task was travel times forecasting. This task was described in Chapter 9. The method described in that chapter optimized only the set of inputs for SVR. We tried to further improve it by simultaneous optimization of the sensor selection and SVR meta-parameters. To compare the prediction provided by these two approaches, we used the data from June 2011 of Seattle senses and Arterial travel times datasets.

The results are compared in Table 10.4, „place begin“ is the identifier of the start of road segment and „place end“ is the identifier of the end of the road segment. The column „use history“ informs whether only currently measured values were used for prediction (-), or a short history (15 min.) was also considered. The results for the method which optimizes only the subset of input sensors are given in column „Only Feature“ and the results for the method which simultaneously optimizes the inputs and meta-parameters are shown in column „Meta-param“. It can be seen in column „Improvement“ that our method provides only a small improvement in this case (the mean is 0.32 %).

Place begin	Place end	Use history	RMSE Only Feature	RMSE Meta-param.	Improvement (%)
58	46	-	57.19	56.99	0.34
58	46	15 min	57.36	57.11	0.43
7	58	-	27.11	26.99	0.44
7	58	15 min	26.12	26.09	0.08

Table 10.4: RMSE for travel times estimation.

10.3 Acceleration of optimization process by parallel implementation

Methods based on genetic algorithms are usually very time consuming. Fortunately, current processors contain many computational cores, which can significantly increase the performance and reduce the time of computation if they are properly used. However, to obtain this higher performance, it is necessary to use algorithms capable of spreading the computational effort among many computational units. We propose a parallel implementation of the method for simultaneous optimization of set of SVR inputs and meta-parameters.

10.3.1 Parallel genetic algorithms

There exist many approaches to parallelize genetic algorithms. The basic approach is known as the master-slave model. In this case, the master process performs all operators like selection, crossover and mutation. The only part of the algorithm which is conducted in parallel, is evaluation of the quality of candidate solutions. In this process, the chromosomes of candidate solutions are distributed to the computational units (slaves), which evaluate the quality of solutions and return computed values of the fitness functions back to the master.

Another approach is called the multiple-population or island model, in which each computational unit has its own population of candidate solutions and performs all genetic operators on its own. After a predefined number of generations the most promising solutions are exchanged among the computational units, simulating thus a migration, and the evolution continues [13].

10.3.2 Parallel implementation

In our work we utilized the master-slave principle. Each computational unit, which is one processor core in our case, computes the values of fitness functions for some portion of the population. The parallel implementation is written in R language for statistical computation. We used R library called „foreach“, which is able to parallelize loops in a code based on the OpenMP technology. The only parallel part of our algorithm is the loop with the evaluation of new candidate solutions.

The main problem is a different time needed for evaluation of candidate solutions. These differences are undesirable in the parallel loops because of synchronization problems. Hence we investigated the influence of SVR meta-parameters and the influence of the number of SVR inputs on the solution evaluation time.

We measured the dependency between the value of regularization coefficient C and the time of SVR training for prediction of traffic variables. The results are shown in Figure 10.4. The left side of the Figure is for linear kernel and the right side is for radial kernel. It can be seen that the time is rapidly growing if the regularization coefficient is higher than 2^3 for linear kernel and 2^7 for radial kernel. Because of it we restricted the values in the chromosomes in the following manner. The C_{linear} gene is restricted to be within the interval $[-5, 3]$, C_{radial} in $[-5, 7]$ and gamma in $[-15, 3]$.

10.3.3 Parallel implementation speedup

In order to show the speedup provided by parallel implementation, we performed 10 independent runs of our method for short time traffic prediction using a different number of

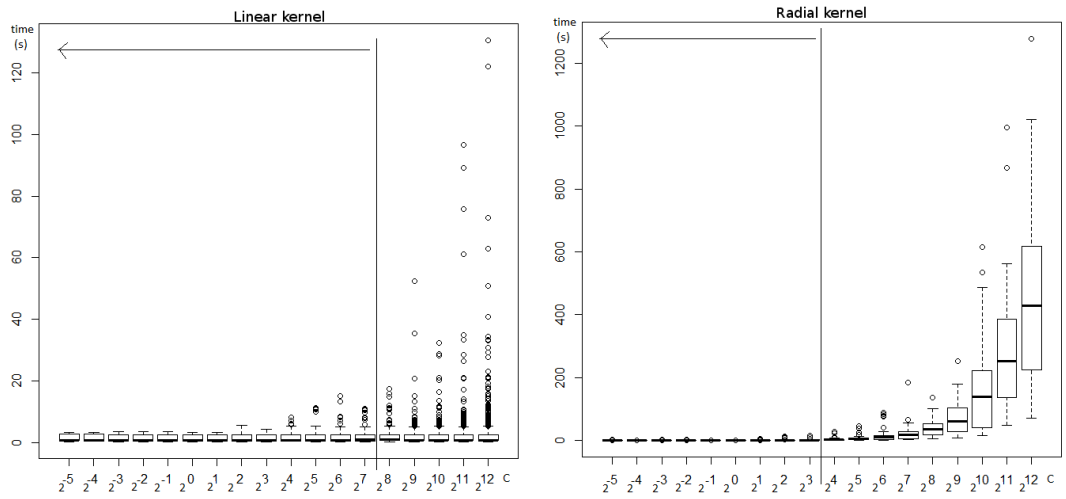

Figure 10.4: Dependency between regularisation coefficient (C) and the time of training.

processor cores. We measured the number of GA iterations (generations) performed in 1 hour.

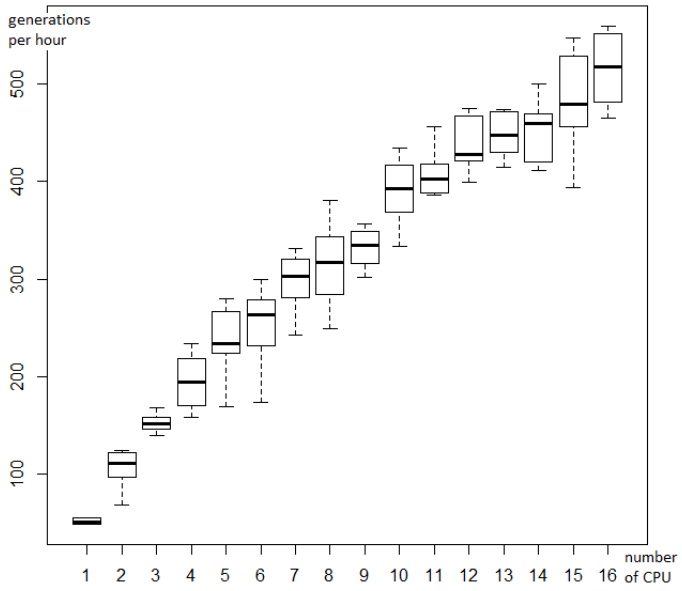

Figure 10.5: Speedup for short time traffic forecasting using parallel version of our method.

The population consists of 40 individuals. According to our experiments (Figure 10.5), it is possible to achieve speedup of almost 10 times when 16 processor cores are employed.

Chapter 11

Discussion and Comparison with Other Methods

In Chapters 8, 9 and 10, we have proposed new methods for data imputation, short time traffic forecasting and estimation of travel times. We have tested these methods using real world data. However, we did not compare them with the other state of the art methods. In this chapter we provide this comparison.

At the begining of this chapter, we will describe practical dificulties with the quality evaluation of machine learning methods and models. Especially the problem of overfitting and underfitting is described. Then we will compare our method for data imputation with methods based on Principal Component Analysis (PCA). In the next part, the results of the method for short time traffic forecasting and travel times prediction which utilize the dynamic model switching will be compared with an approach utilizing only a single SVR and performing the imputation of missing input data using the zero and mean values.

11.1 Evaluation of prediction system quality

In the area of machine learning, it is very important to evaluate the quality of the trained model. It is often non-trivial, because underfitting and overfitting problems can appear. The underfitting problem occurs when the machine learning model is too simple to learn to solve the given problem. The overfitting appears when the model is very complex and it provides very good results for training samples, however it does not generalize well and provides poor results for unseen samples.

To partly avoid these problems, we usually split the available data into two distinct datasets. The first set is used for training and the second for evaluation of the prediction quality. This basic approach works well in simple scenarios. However, sometimes we need to know the prediction quality of the model before its learning is finished. For example, we need it to evaluate the termination condition in iterative algorithms, or we use it in the calibration and optimization of meta-parameters for machine learning algorithm. In this case, we need to split the available data into three groups. The first is utilized for training, the second is used in the termination condition or for calibration and the third is for final evaluation of the obtained model. This approach is called the cross-validation and we will use it in this chapter to compare our methods with other methods [91].

For each of the discused methods, we will split the data into a training set for learning and a test set for evaluation of the objective functions in genetic algorithm. After the model

is finally created, the validation set is used to evaluate the prediction quality. The reason we use the validation set for comparison is that these data were not used for training and during the optimization process and they can provide us a fair comparison.

11.2 Data imputation

The Principal Component Analysis is often used for dimensionality reduction of the input data. However, after some modifications, it can be also utilized to estimate the missing values. In this chapter, we provide the results of the PCA methods for imputation of the missing traffic data and compare them with the proposed method, which utilizes SVR.

We performed the imputation using two variants of PCA method. The first is called Singular Value Decomposition [131] and the second is probabilistic principal component analysis (PPCA) [105]. The results for the imputation of traffic volume are shown in Table 11.1 - four sensors with the biggest mean volume on four places in the centre of Seattle are analysed. We used the data from 1st July to 15th July as the training set, 16th to 31st July as the test set and the data from August as the validation set. The RMSE in Table 11.1 is given for the validation set.

Place	Sensor	Our Method	PCA method		PPCA method	
		RMSE	RMSE	Improvement	RMSE	Improvement
11	1	6.70	6.46	−3.72 %	6.49	−3.24 %
11	3	6.50	6.93	**6.20 %**	6.70	**2.99 %**
11	13	6.94	6.94	0.00 %	7.02	**1.14 %**
11	17	5.20	5.30	**1.89 %**	5.30	**1.89 %**
19	3	5.98	7.91	**24.40 %**	7.53	**20.58 %**
19	4	6.14	7.30	**15.89 %**	7.23	**15.08 %**
19	7	5.54	7.87	**29.61 %**	7.78	**28.79 %**
19	8	4.93	5.93	**16.86 %**	5.82	**15.29 %**
22	1	4.20	5.12	**17.97 %**	4.90	**14.29 %**
22	4	5.44	6.73	**19.17 %**	6.64	**18.07 %**
22	7	4.91	5.40	**9.07 %**	5.40	**9.07 %**
22	8	4.35	5.42	**19.74 %**	5.18	**16.02 %**
23	6	4.43	4.89	**9.41 %**	4.84	**8.47 %**
23	7	11.27	11.39	**1.05 %**	10.48	−7.54 %
23	8	10.95	10.86	−0.83 %	9.95	−10.05 %
23	9	5.07	5.98	**15.22 %**	5.83	**13.04 %**

Table 11.1: Results of the volume imputation compared with PCA and PPCA method (Seattle Sensys data).

It can be observed that our method provides a better precision for almost every sensors. The exception are sensors number 1 on place 11 and number 8 on place 23, where Singular Value Decomposition and PPCA perform slightly better. For sensor 7 on place 23, the PPCA method also works better than our method. The results for occupancy under the same test scenario are provided in Table 11.2. In this case, our method based on the SVR is in most cases worse than Singular Value Decomposition (6 sensors out of 15) and PPCA (4 sensors out of 15). This might be caused by highrer noise in occupancy data. The rate between standard deviation and mean value is usually higher than for volume (Figure 6.6).

Place	Sensor	Our Method	PCA method		PPCA method	
		RMSE	RMSE	Improvement	RMSE	Improvement
11	1	7.28	7.72	**5.70 %**	7.70	**5.45 %**
11	3	6.52	6.60	**1.21 %**	6.63	**1.66 %**
11	4	8.41	8.48	**0.83 %**	8.43	**0.24 %**
11	17	7.04	6.74	−4.45 %	6.77	−3.99 %
19	3	3.66	4.17	**12.23 %**	4.17	**12.23 %**
19	4	4.72	4.67	−1.07 %	4.68	−0.85 %
19	8	10.44	9.13	−14.35 %	8.97	−16.39 %
22	1	4.26	4.12	−3.40 %	4.11	−3.65 %
22	3	5.14	4.52	−13.72 %	4.52	−13.72 %
22	6	3.92	3.87	−1.29 %	3.87	−1.29 %
22	8	4.15	3.95	−5.06 %	3.96	−4.80 %
23	1	12.55	12.59	**0.32 %**	12.52	−0.24 %
23	2	10.27	10.25	−0.20 %	10.16	−1.08 %
23	3	11.61	11.45	−1.40 %	11.20	−3.66 %
23	7	21.35	20.89	−2.20 %	20.57	−3.79 %

Table 11.2: Results of the occupancy imputation compared with PCA and PPCA method.

11.3 Short Term Traffic Forecasting

To justify the proposed approach for traffic forecasting which is based on multi-objective genetic algorithms, we will compare it with the method utilizing only SVR. It is almost impossible to utilize SVR directly for traffic forecasting, because there are many missing values in the real world data. This is problematic because the SVR needs all inputs available. Hence, we have to fill these missing input data somehow. In our experiments, we utilized two simple techniques. The first technique fills the missing values by zero and the second one by the mean value of the given variable (computed from the previous samples).

It is important to note that our method based on the multi-objective GA utilizes many SVRs which differ in the input variables involved and these SVRs can be used according to which input data are currently available. Thus, it is not necessary to directly deal with missing data.

The results for traffic forecasting of traffic volume with the prediction horizon of 15 minutes are shown in Table 11.3. We employed the data sets in the same way as in the previous task.

The results have shown that our method performs better in almost all cases. The approach based on a simple SVR with the missing inputs filled with zeros performs slightly better only for one sensor. If the mean is used, 2 out of 15 cases are predicted more precisely.

We tested the same scenario of short term traffic forecasting for the occupancy. In this case, our method provide better prediction than simple SVR with missing inputs filled by zero as well as the mean for 10 out of 15 sensors. The results are shown in Table 11.4.

Place	Sensor	Our Method	Imputation by zero		Imputation by mean	
		RMSE	RMSE	Improvement	RMSE	Improvement
11	1	7.50	8.01	**6.37 %**	8.08	**7.18 %**
11	3	8.12	8.47	**4.13 %**	8.78	**7.52 %**
11	13	7.88	9.10	**13.41 %**	9.85	**20.00 %**
11	17	6.81	9.36	**27.24 %**	10.75	**36.65 %**
19	3	7.76	10.35	**25.02 %**	11.46	**32.29 %**
19	4	7.83	10.98	**28.69 %**	10.65	**26.48 %**
19	7	7.78	9.86	**21.10 %**	11.34	**31.39 %**
19	8	6.64	8.71	**23.77 %**	8.02	**17.21 %**
22	1	6.23	6.35	**1.89 %**	5.72	-8.92 %
22	4	7.59	9.88	**23.18 %**	9.27	**18.12 %**
22	7	6.97	6.81	-2.35 %	7.30	**4.52 %**
22	8	6.46	7.59	**14.89 %**	8.04	**19.65 %**
23	6	6.04	6.95	**13.09 %**	6.01	-0.50 %
23	7	6.39	16.56	**61.41 %**	14.96	**57.29 %**
23	8	6.65	14.86	**55.25 %**	13.67	**51.35 %**
23	9	6.39	10.79	**40.78 %**	9.88	**35.32 %**

Table 11.3: Results of volume forecasting compared with imputation by the zero and mean.

Place	Sensor	Our Method	Imputation by zero		Imputation by mean	
		RMSE	RMSE	Improvement	RMSE	Improvement
11	1	7.10	7.77	**8.62%**	7.80	**8.97 %**
11	17	6.49	8.27	**21.52%**	8.43	**23.01 %**
11	3	5.52	6.54	**15.60%**	6.59	**16.24 %**
11	4	8.76	8.74	- 0.23%	8.64	-1.39 %
19	3	3.80	4.65	**18.28%**	4.80	**20.83 %**
19	4	4.90	5.58	**12.19%**	5.46	**10.26 %**
19	8	9.09	12.65	**28.14%**	12.38	**26.58 %**
22	1	4.82	4.59	- 5.01%	4.60	-4.78 %
22	3	4.90	7.08	**30.79%**	7.28	**32.69 %**
22	6	4.59	4.24	- 8.25%	4.29	-6.99 %
22	8	4.33	4.43	**2.26%**	4.43	**2.26 %**
23	1	14.99	13.38	-12.03%	13.79	-8.70 %
23	2	11.98	11.08	- 8.12%	10.95	-9.41 %
23	3	13.72	13.95	**1.65%**	14.42	**4.85 %**
23	7	20.07	21.78	**7.85%**	22.05	**8.98 %**

Table 11.4: Results of occupancy forecasting compared with imputation by the zero and mean.

11.4 Travel Times Forecasting

The same approach, the same utilization of data and filling the missing values was implemented for the travel times forecasting. The results summarized in Table 11.5 show that our multi-objective method outperforms the simple SVR method in all test scenarios.

Begin	End	Our Method	Imputation by zero		Imputation by mean	
		RMSE	RMSE	Improvement	RMSE	Improvement
Imputation by zero						
Place 7	Place 58	26.04	43.72	**40.44%**	31.92	**18.42%**
Place 58	Place 46	53.42	57.71	**7.43%**	79.74	**33.01%**
Imputation by mean						
Place 7	Place 58	25.68	42.28	**39.26%**	34.69	**25.97%**
Place 58	Place 46	52.99	63.33	**16.33%**	59.78	**5.61%**

Table 11.5: Results of travel times forecasting compared with imputation by the zero and mean.

Chapter 12

Conclusions

The main objective of this thesis was to improve soft-computing methods for road traffic prediction by utilizing multiobjective evolutionary algorithms. In the first chapters, we have formally described traffic variables, which are often subject of measurement and prediction, and summarised the basic principles of machine learning algorithms and genetic algorithms. A special focus was given to support vector regression and multi-objective genetic algorithms. Then we provided a brief history of road traffic prediction and described the state of the art methods in this area.

We analyzed available road traffic data, especially the data provided by Research Data Exchange project. The results of these analyses revealed, that the modern prediction methods must be able to deal with many missing values and outliers if they should be successful. Moreover, the results have shown that it is usefull to combine the input data coming from varisous sources, such as traffic sensors or traffic cameras. This analysis helped us to define the goals of this thesis.

The first goal of the thesis was to create a general framework for traffic prediction and travel times estimation. This framework should internally use a multi-objective optimization (such as evolutionary algorithms) in order to provide good trade-offs between various conflicting objectives the users typically formulate in this domain. Moreover, this framework should be capable of working with real world traffic data, which often contain a huge portion of missing values.

To fulfill this goal, we proposed a prediction framework, which internally utilizes SVR-based prediction. The meta-parameters and inputs of SVR are simultaneously optimized by a multi-objective genetic algorithm. The multimodal-NSGAII algorithm is used to perform this optimization task. We choose the RMSE of prediction, the number of SVR inputs and the portion of time in which the SVR can not be used for prediction (because of missing data) as the objective functions. The multi-objective optimization provided us with many solutions (SVRs), which differ in values of the objective functions. We typically obtained solutions showing very small prediction errors (RMSE), but requiring many input variables, but they often cannot be used because of missing input data. On the other hand, we obtained solutions with a higher prediction error which utilize only a few input variables. We also got many compromise solutions between these two extremes.

The reason why it is good to have multiple models with different characteristics is that we can switch among them during the real time prediction process. The main factor which is changing over the time is the data availability from sensors or camera. The key idea of the approach developed in this thesis is to dynamicaly switch among these models according to which data are currently available. The highest-quality model is activated if possible.

If the input data for this model are missing, the second best model is taken. If the input data for this model are not available, we will use the next model. We can continue with this process, until we get the model for which the data are available, or we have to stop because the data are unavailable.

The second goal of this thesis was to evaluate this framework on real world case studies. In the first case study, we improved an existing approach to estimation of missing values in traffic density maps by employing a multi-objective genetic algorithm (Chapter 7). In the second case study, which was described in Chapter 8, we have shown that it is possible to use multi-objective optimization to obtain different SVR models which differ in the prediction error and the number of input sensors. Obtained models were successfully used for data imputation and for short term traffic prediction. It appears that the single objective optimization algorithm cannot deal with this task because it opportunistically prefers SVRs, which are very precise, but require many inputs. The SVRs obtained by a single objective optimization can not be used in practice because of many missing values in the input data.

In the third case study, described in Chapter 9, we utilized our framework to estimate the travel times. In this study, we firstly used the method for dynamic switching of SVR according to available data. We compared our method with a method which uses only a license plate data on two road segments in the centre of Seattle. The proposed method provided more precise prediction in both cases.

In previous two test scenarios we optimized only a set of inputs for SVRs, but the SVR meta-parameters remained constant. However, this is far from the optimum. In Chapter 10, we performed experiments in which SVR inputs and meta-parameters were simultaneously optimized. We evaluated this approach for three different tasks. It was shown that the optimization of meta-parameters can significantly improve the prediction quality. Moreover, we implemented and tested a parallel implementation of our method for this case study. The biggest problem with this implementation was the different time of SVR learning for different meta-parameters. We solved this issue by restricting the meta-parameters values to a predefined range. Our experiments revealed that the parallel implementation is scaling almost linearly with the number of cores.

In the last case study, we compared our prediction framework with the current state of the art methods. In order to provide a fair comparison, we utilized a validation data set containing new data which were used neither during learning nor the optimization process. Results were provided for three different tasks: data imputation, short term traffic forecasting and estimating of travel times. In the case of data imputation, we compared our framework with Singular Value Decomposition and Probabilistic Principal Component Analysis. The results shown that our method provides better results than these methods in the imputation of volume and provides slightly worse results for the imputation of occupancy.

In the case of short time traffic prediction, we have compared our method with a single SVR. The missing inputs of this SVR were filled by zero or the mean values of the missing variable. In this case, our multi-objective method performed better than simple SVR in almost all cases. The last task was the estimation of travel times. In this case, we also used a simple SVR with the missing input values filled with zero or the mean. Here, our multi-objective method outperformed the single SVR for all test road segments.

The main disadvantage of our prediction framework is its speed. The GA needs to evaluate many candidate solutions, which is often very time consuming. We have proposed a basic parallel implementation in this thesis. However, the modern hardware such as Intel Xeon Phi acceleration cards could possibly provide much better performance [60]. It is also

possible to use modern techniques in evolutionary algorithms such as coevolution to reduce the optimization time [116]. We use only SVR to predict the traffic variables. However, there are many other machine learning models such as neural networks and others. The usage of these models in our framework could lead to an additional improvement in the quality of prediction.

Appendix A: Prediction Results

This appendix contains examples of the results which were produced by our prediction framework. The results are shown in the form of graphs, where the x-axis represents time and y-axis axis represents the variable of interests. The black line shows the real values measured by sensor or camera. The red line represents the values produced by our framework. The following examples are covered in this chapter:

- data imputation provided by different SVRs which were obtained at the end of one run of a multi-objective genetic algorithm,
- traffic forecasting provided by different SVRs which were obtained at the end of one run of a multi-objective genetic algorithm,
- of data imputation obtained using our prediction framework,
- of traffic forecasting produced by our prediction framework and
- and the example of travel times estimation provided by our prediction framework.

12.1 Data imputation provided by different compromise solutions (SVRs) obtained by multi-objective GA

- An example of the data imputation for traffic volume provided by different compromise SVRs for sensor 4 on place 19 is shown in Figure 12.1. These SVRs were obtained at the end of one run of a multi-objective GA. The results are provided for the whole test set.
- The results were obtained according to Chapter 8. Black line is missing in some parts because of a failure of a corresponding equipment.

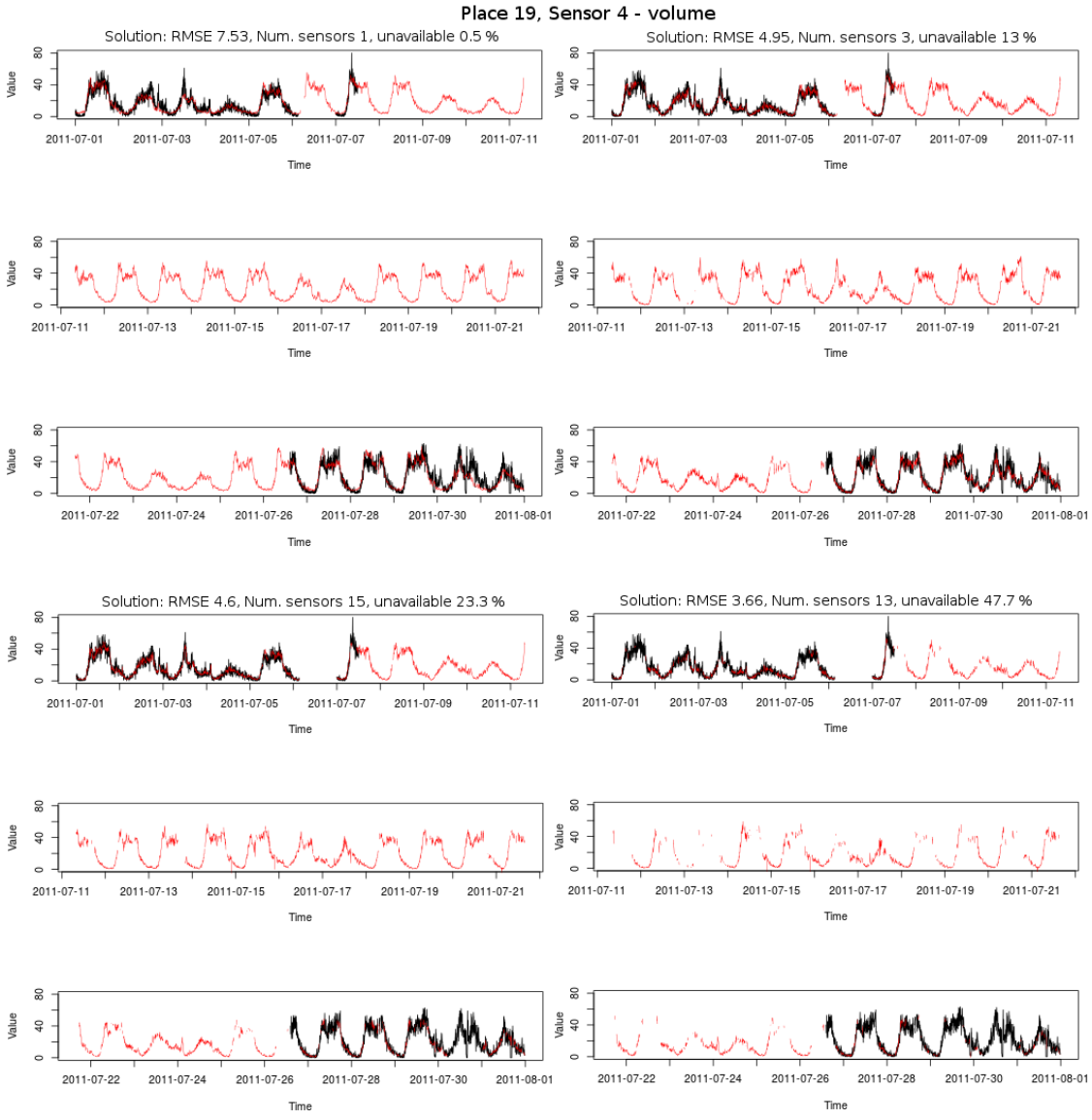

Figure 12.1: Data imputation provided by different compromise SVRs for sensor 4 on place 19. Red line - imputed values, black line - correct values. The values are given as the number of vehicles per five minutes.

- An example of data imputation for volume provided by different SVRs for sensor 4 on place 22 given in Figure 12.2. These SVRs were obtained at the end of one run of a multi-objective GA.
- Detailed results for one week are shown.
- The results were obtained according to Chapter 8.

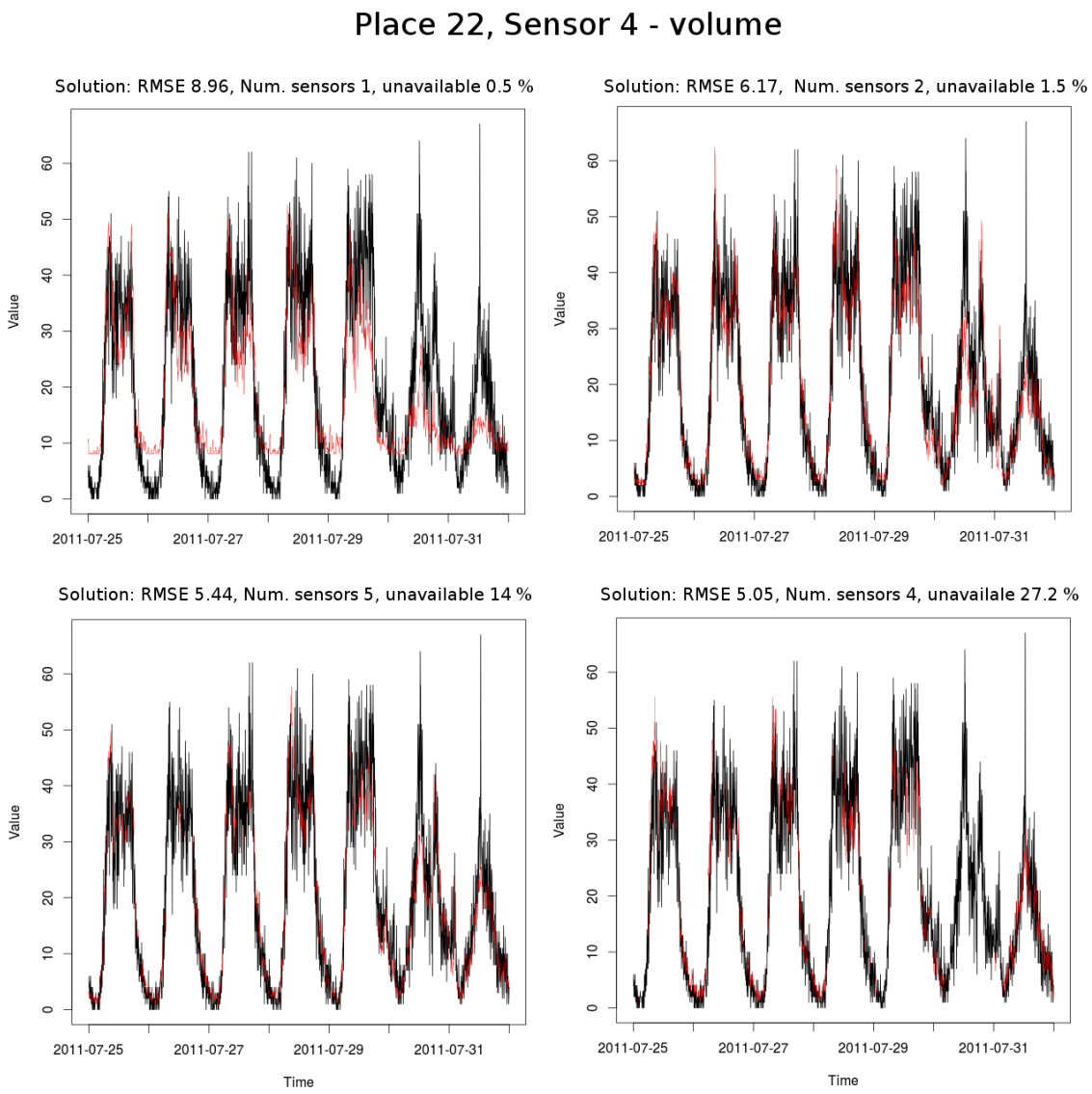

Figure 12.2: Data imputation provided by different compromise SVRs for sensor 4 on place 22. Red line - imputed values, black line - correct values. The results are given as the number of vehicles per five minutes.

12.2 Traffic forecasting provided by different compromise solutions (SVRs) obtained by multi-objective GA

- An example of traffic volume forecasting with horizon of 15 minutes provided by different SVRs for sensor 4 on place 19 is shown in Figure 12.3. These SVRs were obtained at the end of one run of a multi-objective GA. The results are provided for the whole test set.
- The results were obtained according to Chapter 8. Black line is missing in some parts because of a failure of a corresponding equipment.

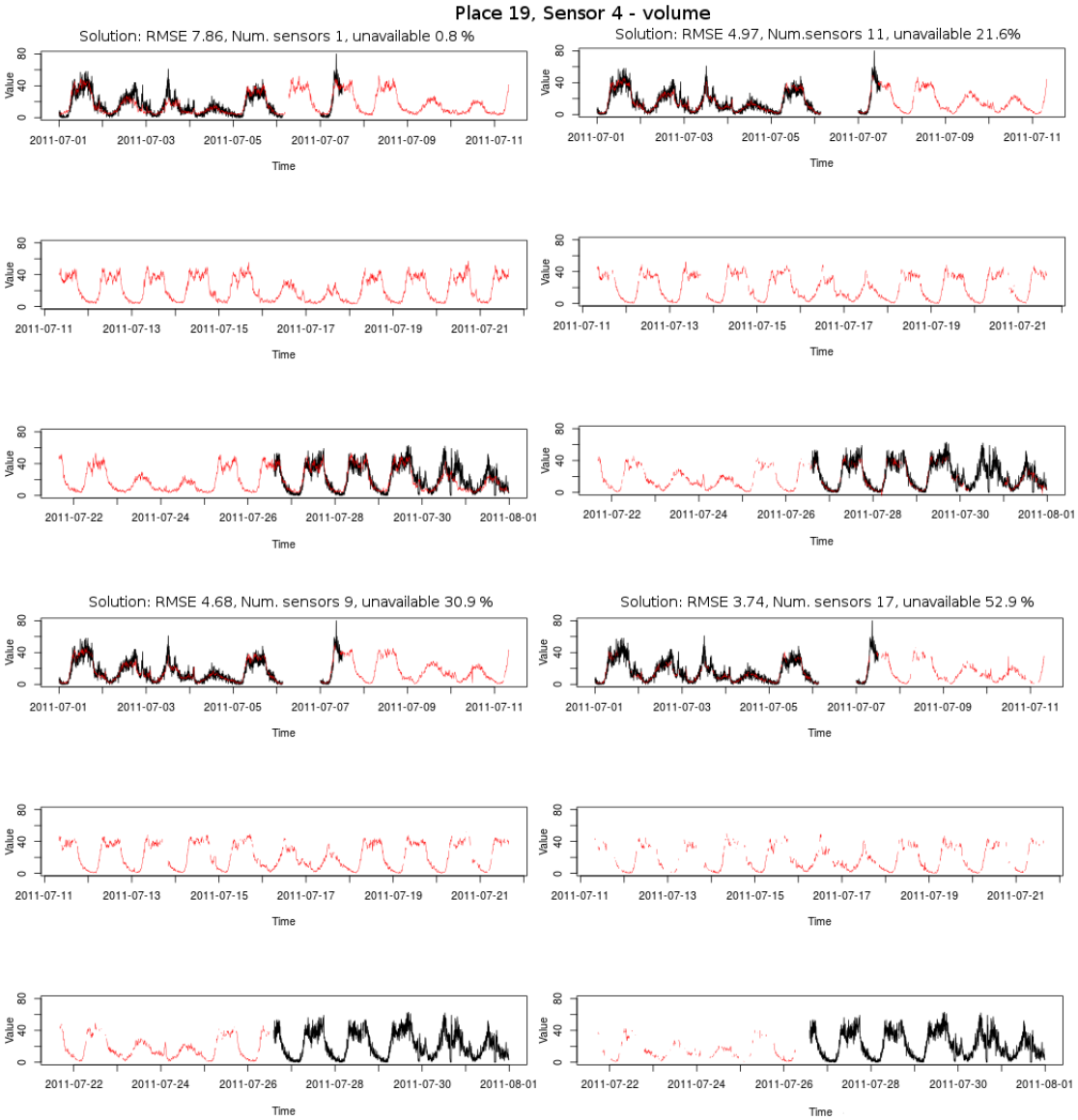

Figure 12.3: Traffic forecasting provided by different compromise SVRs for sensor 4 on place 19 is shown in Figure . Red line - predicted values, black line - correct values. The values are given as the number of vehicles per five minutes.

- An example of traffic volume forecasting provided by different SVRs for sensor 4 on place 22 is given in Figure 12.4. These SVRs were obtained at the end of one run of a multi-objective GA.
- Detailed results for one week are shown.
- The results were obtained according to Chapter 8.

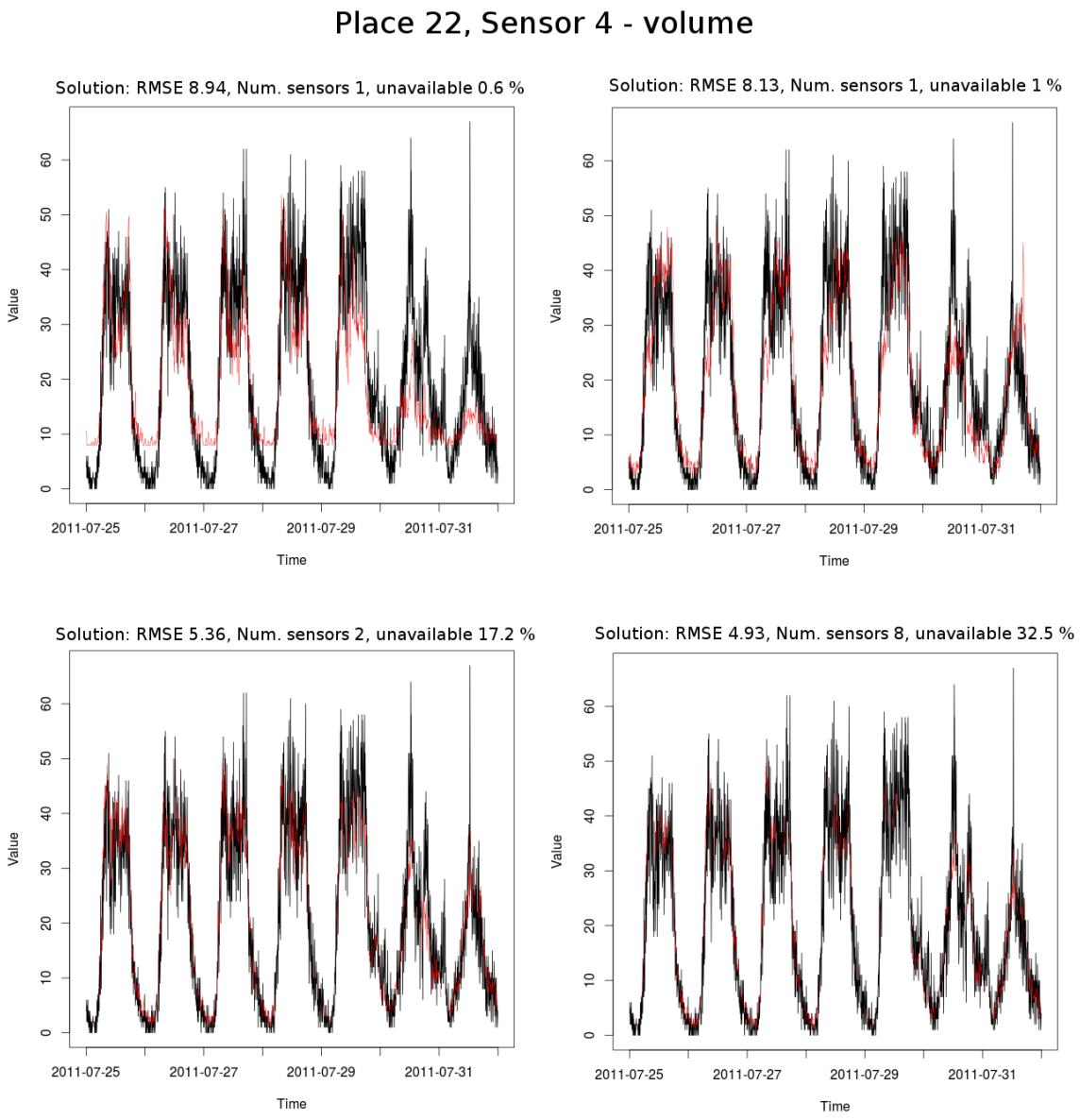

Figure 12.4: Traffic forecasting provided by different compromise SVRs for sensor 4 on place 22. Red line - predicted values, black line - correct values. The values are given as the number of vehicles per five minutes.

12.3 Data imputation

- An example of data imputation for traffic volume. These results were obtained using our prediction framework.
- The results are provided for three days of the validation data set.
- The results were obtained according to Chapter 11.

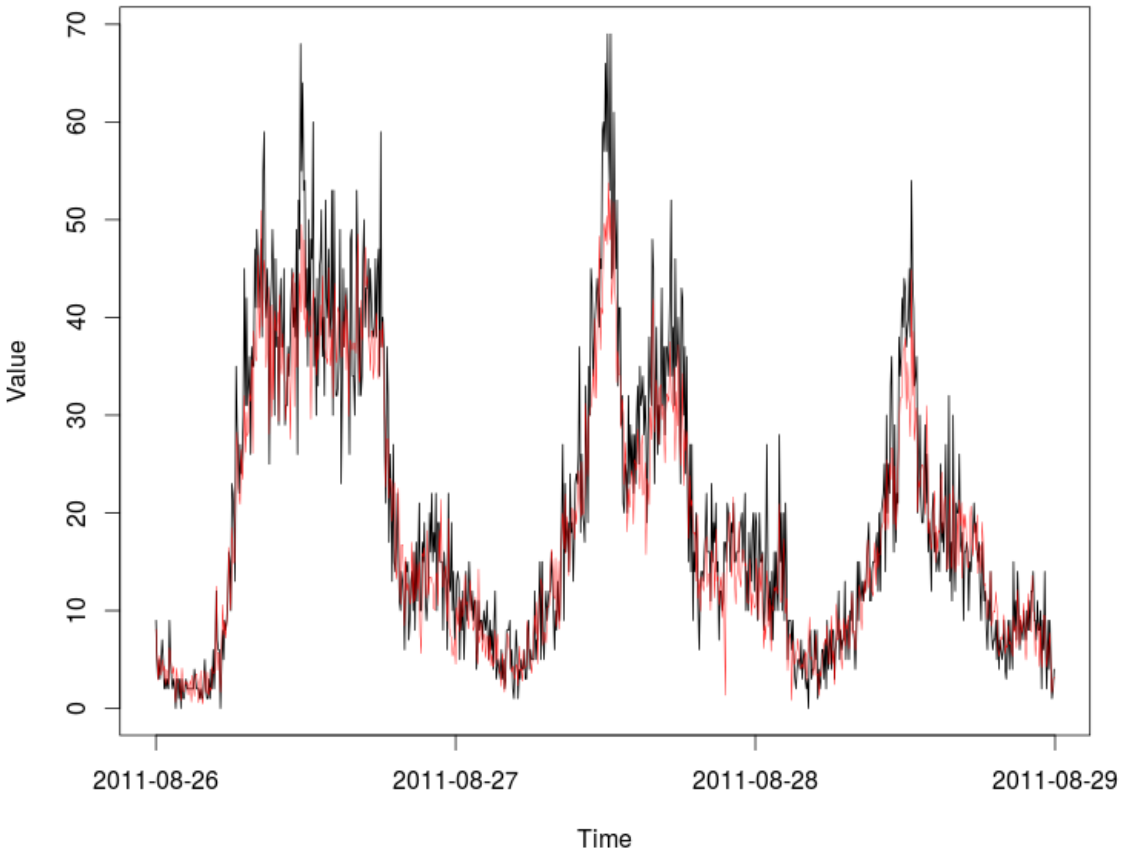

Figure 12.5: Results for data imputation for sensor 4 on place 22. Red line - imputed values, black line - correct values. The values are given as the number of vehicles per five minutes.

12.4 Short-term traffic forecasting

- An example of short term traffic forecasting for traffic volume. The prediction horizon is 15 minutes. These results were obtained using our prediction framework.
- The results are provided for three days of the validation set.
- The results were obtained according to Chapter 11.

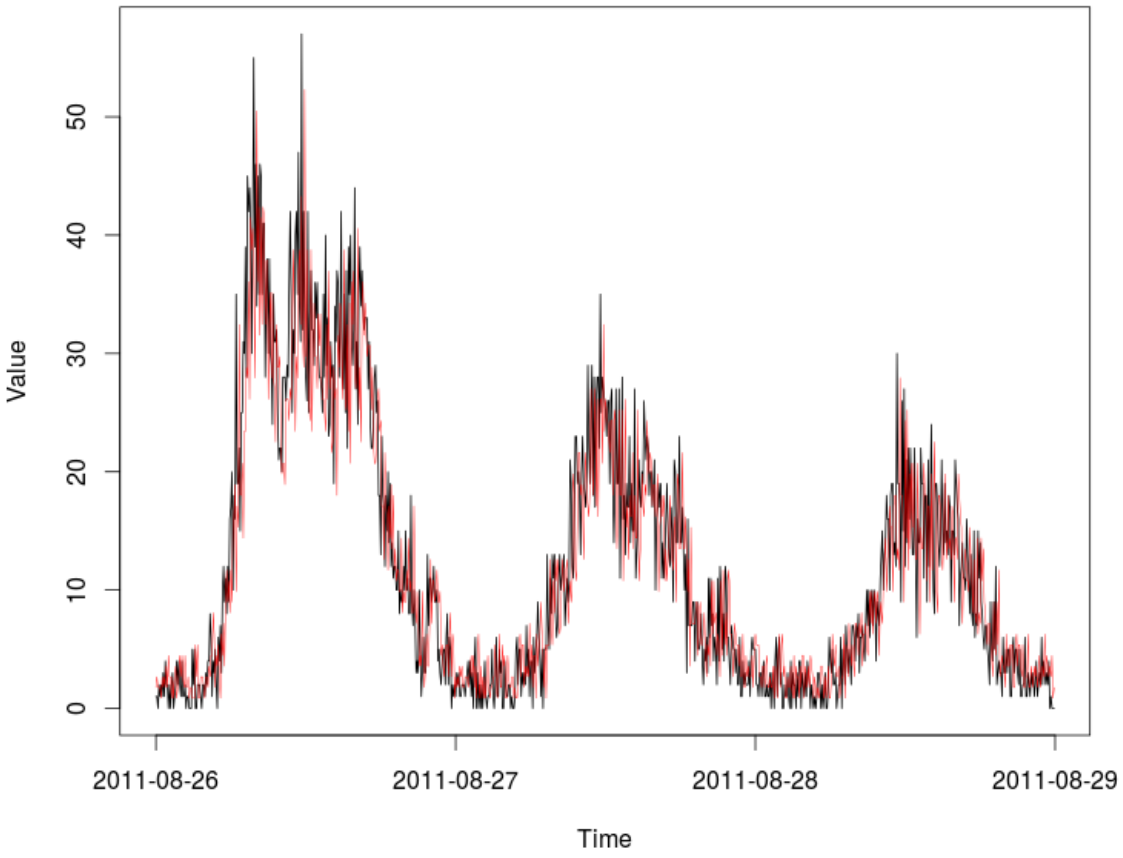

Figure 12.6: Results for traffic forecasting for sensor 4 on place 22. Red line - predicted values, black line - correct values. The values are given as the number of vehicles per five minutes.

12.5 Estimation of travel times

- An example of estimation of travel times. These results were obtained using our prediction framework.
- The results are provided for three days of the validation data set.
- The results were obtained according to Chapter 11.

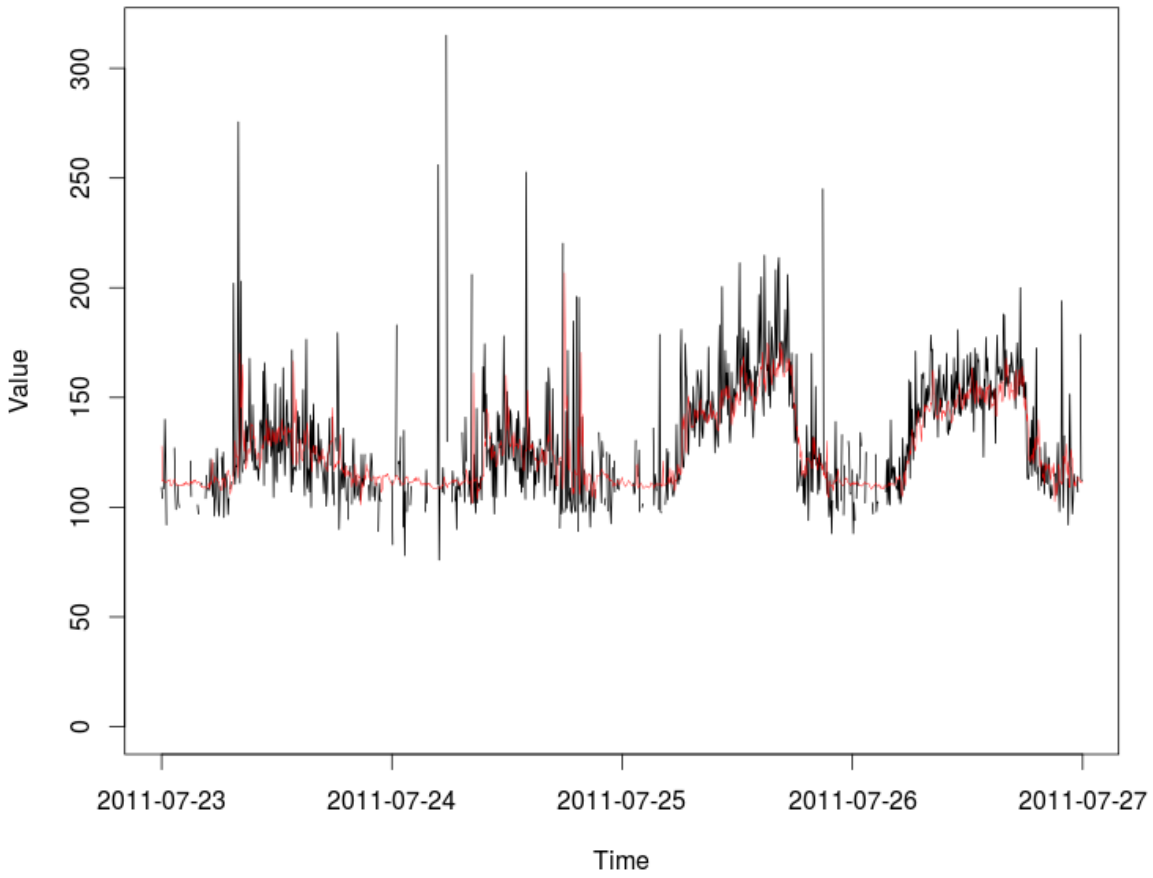

Figure 12.7: Results for travel times estimation for a section starting at place 7 and ending at place 58. Red line - estimated values, black line - correct values. The values are in seconds.

Appendix B: Publications and products

Publications

- **Petrlik Jiri, Korcek Pavol, Fucik Otto, Beszedes Marian and Sekanina Lukas. Estimation of traffic density map using evolutionary algorithm. In: Proceedings of the 15th International IEEE Conference on Intelligent Transportation Systems. Anchorage: IEEE Intelligent Transportation Systems Society, 2012, pp. 632-637. ISBN 978-1-4673-3062-6**
- Petrlik Jiri and Sekanina Lukas. Multiobjective Evolution of Multiple-Constant Multipliers. In: Proceedings of the 18th International Conference on Soft Computing (MENDEL2012). Brno: Faculty of Mechanical Engineering BUT, 2012, pp. 64-69. ISBN 978-80-214-4540-6.
- Petrlik Jiri and Sekanina Lukas. Multiobjective evolution of approximate multiple constant multipliers. In: IEEE International Symposium on Design and Diagnostics of Electronic Circuits and Systems 2013. Brno: IEEE Computer Society, 2013, pp. 116-119. ISBN 978-1-4673-6133-0.
- **Petrlik Jiri, Fucik Otto and Sekanina Lukas. Multiobjective Selection of Input Sensors for Travel Times Forecasting Using Support Vector Regression. In: 2014 IEEE Symposium on Computational Intelligence in Vehicles and Transportation Systems Proceedings. Piscataway: Institute of Electrical and Electronics Engineers, 2014, pp. 14-21. ISBN 978-1-4799-4498-9.**
- **Petrlik Jiri, Fucik Otto and Sekanina Lukas. Multiobjective Selection of Input Sensors for SVR Applied to Road Traffic Prediction. In: Parallel Problem Solving from Nature - PPSN XIII. Heidelberg: Springer Verlag, 2014, pp. 802-811. ISBN 978-3-319-10761-5.**
- **Petrlik Jiri and Sekanina Lukas. Towards Robust and Accurate Traffic Prediction Using Parallel Multiobjective Genetic Algorithms and Support Vector Regression. In: 2015 IEEE 18th International Conference on Intelligent Transportation Systems. Los Alamitos: IEEE Computer Society, 2015, pp. 2231-2236. ISBN 978-1-4673-6596-3.**

Software

- Petrlik Jiri: SVM Feature Selection System, software, 2013

Research Projects and Grants

- Natural Computing on Unconventional Platforms, GACR, GAP103/10/1517, 2010-2013, completed
- The IT4Innovations Centre of Excellence, MSMT, ED1.1.00/02.0070, 2011-2015, running
- Verification and Optimization of Computer Systems, VUT Brno, FIT-S-12-1, 2012-2014, completed
- Research and development focused on monitoring and management of lorry movement on lower class road network in the Czech Republic, TACR, TA02030841, 2012-2014, completed
- Architecture of parallel and embedded computer systems, VUT Brno, FIT-S-14-2297, 2014-2016, running
- Advanced Methods for Evolutionary Design of Complex Digital Circuits, GACR, GA14-04197S, 2014-2016, running
- Verification of the implementation of continuous traffic load map using modern classification and prediction methods, TACR, TA02030915, 2012-2014, completed

Bibliography

[1] Datex II. `http://www.datex2.eu/`. Accessed: 2015-10-20.

[2] RDE Home Page. `http://www.its-rde.net/`. Accessed: 2014-10-5.

[3] Kalidas Ashok and Moshe E Ben-Akiva. Dynamic origin-destination matrix estimation and prediction for real-time traffic management systems. In *International Symposium on the Theory of Traffic Flow and Transportation (12th: 1993: Berkeley, Calif.). Transportation and traffic theory*, 1993.

[4] Thomas Back and Martin Schutz. Intelligent mutation rate control in canonical genetic algorithms. In ZbigniewW. Ras and Maciek Michalewicz, editors, *Foundations of Intelligent Systems*, volume 1079 of *Lecture Notes in Computer Science*, pages 158–167. Springer Berlin Heidelberg, 1996.

[5] Debasish Basak, Srimanta Pal, and Dipak Chandra Patranabis. Support vector regression. *Neural Information Processing-Letters and Reviews*, 11(10):pages 203–224, 2007.

[6] Moshe Ben-Akiva, Michel Bierlaire, Didier Burton, Haris N Koutsopoulos, and Rabi Mishalani. Network state estimation and prediction for real-time traffic management. *Networks and Spatial Economics*, 1(3-4):pages 293–318, 2001.

[7] Rahim F Benekohal. Procedure for validation of microscopic traffic flow simulation models. 12(1320):pages 3–23, 1991.

[8] Sharminda Bera and KV Rao. Estimation of origin-destination matrix from traffic counts: the state of the art. 2011.

[9] Michel Bierlaire and Frank Crittin. An efficient algorithm for real-time estimation and prediction of dynamic od tables. *Operations Research*, 52(1):pages 116–127, 2004.

[10] Stephen Boyd and Lieven Vandenberghe. *Convex optimization*. Cambridge university press, 2004. ISBN: 9780521833783.

[11] Christopher JC Burges. A tutorial on support vector machines for pattern recognition. *Data mining and knowledge discovery*, 2(2):pages 121–167, 1998.

[12] Francesco Camastra. Data dimensionality estimation methods: a survey. *Pattern recognition*, 36(12):pages 2945–2954, 2003.

[13] Erick Cantú-Paz. A survey of parallel genetic algorithms. *Calculateurs paralleles, reseaux et systems repartis*, 10(2):pages 141–171, 1998.

[14] Lijuan Cao. Support vector machines experts for time series forecasting. *Neurocomputing*, 51:pages 321–339, 2003.

[15] Manoel Castro-Neto, Young-Seon Jeong, Myong-Kee Jeong, and Lee D Han. Online-svr for short-term traffic flow prediction under typical and atypical traffic conditions. *Expert systems with applications*, 36(3):pages 6164–6173, 2009.

[16] Robert E Chandler, Robert Herman, and Elliott W Montroll. Traffic dynamics: studies in car following. *Operations research*, 6(2):pages 165–184, 1958.

[17] Sing Yiu Cheung, Sinem Coleri Ergen, and Pravin Varaiya. Traffic surveillance with wireless magnetic sensors. In *Proceedings of the 12th ITS world congress*, volume 1917, pages 173–181, 2005.

[18] Mashrur Chowdhury and Adel W Sadek. Advantages and limitations of artificial intelligence. *Artificial Intelligence Applications to Critical Transportation Issues*, 6:pages 6–8, 2012.

[19] Daniel J Dailey, Fritz W Cathey, and Suree Pumrin. An algorithm to estimate mean traffic speed using uncalibrated cameras. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):pages 98–107, 2000.

[20] Gary A Davis, Nancy L Nihan, Mohammad M Hamed, and Leslie N Jacobson. Adaptive forecasting of freeway traffic congestion. *Transportation research record*, (1287):pages 29–33, 1990.

[21] Kalyanmoy Deb. *Multi-objective optimization using evolutionary algorithms*, volume 16. John Wiley & Sons, 2001. ISBN: 9780471873396.

[22] Kalyanmoy Deb and Ram B Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(3):pages 1–15, 1994.

[23] Kalyanmoy Deb and Hans-Georg Beyer. Self-adaptive genetic algorithms with simulated binary crossover. *Evolutionary Computation*, 9(2):pages 197–221, 2001.

[24] Kalyanmoy Deb and Tushar Goel. Controlled elitist non-dominated sorting genetic algorithms for better convergence. In *Evolutionary Multi-Criterion Optimization*, pages 67–81. Springer, 2001.

[25] Kalyanmoy Deb and Himanshu Jain. An evolutionary many-objective optimization algorithm using reference-point-based nondominated sorting approach, part i: Solving problems with box constraints. *IEEE Transactions on Evolutionary Computation*, 18(4):pages 577–601, 2014.

[26] Kalyanmoy Deb, Amrit Pratap, Sameer Agarwal, and TAMT Meyarivan. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):pages 182–197, 2002.

[27] Kalyanmoy Deb and A Raji Reddy. Reliable classification of two-class cancer data using evolutionary algorithms. *BioSystems*, 72(1):pages 111–129, 2003.

[28] Abhijit Dharia and Hojjat Adeli. Neural network model for rapid forecasting of freeway link travel time. *Engineering Applications of Artificial Intelligence*, 16(7):pages 607–613, 2003.

[29] Abhijit Dharia and Hojjat Adeli. Neural network model for rapid forecasting of freeway link travel time. *Engineering Applications of Artificial Intelligence*, 16(7-8):607 – 613, 2003.

[30] Ailine Ding, Xangmo Zhao, and LiCheng Jiao. Traffic flow time series prediction based on statistics learning theory. In *IEEE 5th International Conference on Intelligent Transportation Systems*, pages 727–730. IEEE, 2002.

[31] Lili Du, Srinivas Peeta, and Yong Hoon Kim. An adaptive information fusion model to predict the short-term link travel time distribution in dynamic traffic networks. *Transportation Research Part B: Methodological*, 46(1):pages 235–252, 2012.

[32] Kaibo Duan, S Sathiya Keerthi, and Aun Neow Poo. Evaluation of simple performance measures for tuning svm hyperparameters. *Neurocomputing*, 51:pages 41–59, 2003.

[33] Patrick AM Ehlert and Leon JM Rothkrantz. Microscopic traffic simulation with reactive driving agents. In *Proceedings Intelligent Transportation Systems*, pages 860–865. IEEE, 2001.

[34] Jan Fabian Ehmke, Stephan Meisel, and Dirk Christian Mattfeld. Floating car based travel times for city logistics. *Transportation research part C: emerging technologies*, 21(1):pages 338–352, 2012.

[35] A.E. Eiben, R. Hinterding, and Z. Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3(2):pages 124–141, Jul 1999.

[36] L Florio and L Mussone. Neural-network models for classification and forecasting of freeway traffic flow stability. *Control Engineering Practice*, 4(2):pages 153–164, 1996.

[37] Jerome Friedman, Trevor Hastie, and Robert Tibshirani. *The elements of statistical learning*, volume 1. Springer series in statistics Springer, Berlin, 2001. ISBN: 9780387848570.

[38] Denos C Gazis, Robert Herman, and Renfrey B Potts. Car-following theory of steady-state traffic flow. *Operations research*, 7(4):pages 499–505, 1959.

[39] Michel Gendreau and Jean-Yves Potvin. *Handbook of metaheuristics*, volume 2. Springer, 2010. 9781441916655.

[40] Nikolas Geroliminis and Jie Sun. Properties of a well-defined macroscopic fundamental diagram for urban traffic. *Transportation Research Part B: Methodological*, 45(3):pages 605–617, 2011.

[41] Olaf Gietelink, Jeroen Ploeg, Bart De Schutter, and Michel Verhaegen. Development of advanced driver assistance systems with vehicle hardware-in-the-loop simulations. *Vehicle System Dynamics*, 44(7):pages 569–590, 2006.

[42] John F Gilmore and Naohiko Abe. Neural network models for traffic control and congestion prediction. *Journal of Intelligent Transportation Systems*, 2(3):pages 231–252, 1995.

[43] Peter G Gipps. A behavioural car-following model for computer simulation. *Transportation Research Part B: Methodological*, 15(2):pages 105–111, 1981.

[44] BD Greenshields, Ws Channing, Hh Miller, et al. A study of traffic capacity. In *Highway research board proceedings*, volume 1935. National Research Council (USA), Highway Research Board, 1935.

[45] Mohammad M Hamed, Hashem R Al-Masaeid, and Zahi M Bani Said. Short-term prediction of traffic volume in urban arterials. *Journal of Transportation Engineering*, 121(3):pages 249–254, 1995.

[46] Jiawei Han, Micheline Kamber, and Jian Pei. *Data mining: concepts and techniques: concepts and techniques*. Elsevier, 2011. ISBN: 9780123814791.

[47] Dirk Helbing. Traffic and related self-driven many-particle systems. *Reviews of modern physics*, 73(4):pages 1067, 2001.

[48] Robert Herman, Elliott W Montroll, Renfrey B Potts, and Richard W Rothery. Traffic dynamics: analysis of stability in car following. *Operations research*, 7(1):pages 86–106, 1959.

[49] R. Hinterding. Gaussian mutation and self-adaption for numeric genetic algorithms. In *IEEE International Conference on Evolutionary Computation.*, volume 1, page 384. IEEE, Nov 1995.

[50] Antoine G Hobeika and Chang Kyun Kim. Traffic-flow-prediction systems based on upstream traffic. In *Vehicle Navigation and Information Systems Conference, 1994. Proceedings., 1994*, pages 345–350. IEEE, 1994.

[51] Wei-Chiang Hong. Traffic flow forecasting by seasonal svr with chaotic simulated annealing algorithm. *Neurocomputing*, 74(12):pages 2096–2107, 2011.

[52] Serge P Hoogendoorn and Piet HL Bovy. State-of-the-art of vehicular traffic flow modelling. *Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering*, 215(4):pages 283–303, 2001.

[53] Maarten Houbraken, Pieter Audenaert, Didier Colle, Mario Pickavet, Karolien Scheerlinck, Isaak Yperman, and Steven Logghe. Real-time traffic monitoring by fusing floating car data with stationary detector data. In *Models and Technologies for Intelligent Transportation Systems (MT-ITS), 2015 International Conference on*, pages 127–131. IEEE, 2015.

[54] Cheng-Lung Huang and Chieh-Jen Wang. A ga-based feature selection and parameters optimizationfor support vector machines. *Expert Systems with applications*, 31(2):pages 231–240, 2006.

[55] Jeffrey Huang, Xuhui Shao, and Harry Wechsler. Face pose discrimination using support vector machines (svm). In *Pattern Recognition, 1998. Proceedings. Fourteenth International Conference on*, volume 1, pages 154–156. IEEE, 1998.

[56] Wei Huang, Yoshiteru Nakamori, and Shou-Yang Wang. Forecasting stock market movement direction with support vector machine. *Computers & Operations Research*, 32(10):pages 2513–2522, 2005.

[57] Aapo Hyvärinen and Erkki Oja. Independent component analysis: algorithms and applications. *Neural networks*, 13(4):pages 411–430, 2000.

[58] Sherif Ishak and Haitham Al-Deek. Statistical evaluation of interstate 4 traffic prediction system. *Transportation Research Record: Journal of the Transportation Research Board*, (1856):pages 16–24, 2003.

[59] Himanshu Jain and Kalyanmoy Deb. An evolutionary many-objective optimization algorithm using reference-point based nondominated sorting approach, part ii: Handling constraints and extending to an adaptive approach. *IEEE Transactions on Evolutionary Computation*, 18(4):pages 602–622, 2014.

[60] James Jeffers and James Reinders. *Intel Xeon Phi coprocessor high-performance programming*. Newnes, 2013. ISBN: 9780124104143.

[61] Shyr-Long Jeng, Wei-Hua Chieng, and Hsiang-Pin Lu. Estimating speed using a side-looking single-radar vehicle detector. *IEEE Transactions on Intelligent Transportation Systems*, 15(2):pages 607–614, 2014.

[62] Ian Jolliffe. *Principal component analysis*. Wiley Online Library, 2002. ISBN: 9780387954424.

[63] K. Kanayama, Y. Fujikawa, K. Fujimoto, and M. Horino. Development of vehicle-license number recognition system using real-time image processing and its application to travel-time measurement. In *Vehicular Technology Conference, 1991. Gateway to the Future Technology in Motion., 41st IEEE*, pages 798–804. IEEE, May 1991.

[64] MG Karlaftis and EI Vlahogianni. Statistical methods versus neural networks in transportation research: Differences, similarities and some insights. *Transportation Research Part C: Emerging Technologies*, 19(3):pages 387–399, 2011.

[65] Jesse H Katz. Simulation of a traffic network. *Communications of the ACM*, 6(8):pages 480–486, 1963.

[66] James H Kell, Iris J Fullerton, and Milton K Mills. *Traffic detector handbook*. 1990. ISBN: 9781420067187.

[67] Arne Kesting. *Traffic Flow Dynamics: Data, Models and Simulation*. Springer, 2012. ISBN: 9783642324604.

[68] Hemanshu S Khatri and Sunil B Somani. Infrared-based system for vehicle counting and classification. In *Pervasive Computing (ICPC), 2015 International Conference on*, pages 1–5. IEEE, 2015.

[69] Howard R Kirby, Susan M Watson, and Mark S Dougherty. Should we use neural networks or statistical models for short-term motorway traffic forecasting. *International Journal of Forecasting*, 13(1):pages 43–50, 1997.

[70] Lawrence A Klein, Milton K Mills, and David RP Gibson. *Traffic Detector Handbook: -Volume II*. 2006.

[71] Dieter Koller, J Weber, T Huang, J Malik, G Ogasawara, B Rao, and S Russell. Towards robust automatic traffic scene analysis in real-time. In *Pattern Recognition, 1994. Vol. 1-Conference A: Computer Vision & Image Processing., Proceedings of the 12th IAPR International Conference on*, volume 1, pages 126–131. IEEE, 1994.

[72] Pavol Korcek, Lukas Sekanina, and Otto Fucik. Evolutionary approach to calibration of cellular automaton based traffic simulation models. In *15th International IEEE Conference on Intelligent Transportation Systems (ITSC),*, pages 122–129. IEEE, 2012.

[73] Pavol Korcek, Lukas Sekanina, and Otto Fucik. Advanced approach to calibration of traffic microsimulation models using travel times. *J. Cellular Automata*, 8(5-6):pages 457–467, 2013.

[74] Hans-Peter Kriegel, Matthias Renz, Matthias Schubert, and Andreas Zufle. Efficient traffic density prediction in road networks using suffix trees. *KI - Kunstliche Intelligenz*, 26(3):pages 233–240, 2012.

[75] Corinne Ledoux. An urban traffic flow model integrating neural networks. *Transportation Research Part C: Emerging Technologies*, 5(5):pages 287–300, 1997.

[76] Clyde E Lee, Glenn E Grayson, Charlie R Copeland, Jeff W Miller, Thomas W Rioux, and Vivek S Savur. The Texas model for intersection traffic: User's guide. Technical report, Center for Highway Research, University of Texas at Austin, 1977.

[77] H Lenz, CK Wagner, and R Sollacher. Multi-anticipative car-following model. *The European Physical Journal B-Condensed Matter and Complex Systems*, 7(2):pages 331–335, 1999.

[78] Moshe Levin and Yen-Der Tsao. On forecasting freeway occupancies and volumes (abridgment). *Transportation Research Record*, (773), 1980.

[79] Lefei Li, W-H Lin, and Hongchao Liu. Type-2 fuzzy logic approach for short-term traffic forecasting. In *IEEE Proceedings-Intelligent Transport Systems*, volume 153, pages 33–40. IEEE, 2006.

[80] Ming-Wei Li, Wei-Chiang Hong, and Hai-Gui Kang. Urban traffic flow forecasting using gauss–svr with cat mapping, cloud model and pso hybrid algorithm. *Neurocomputing*, 99:pages 230–240, 2013.

[81] Q Li, T Zhang, and Y Yu. Using cloud computing to process intensive floating car data for urban traffic surveillance. *International Journal of Geographical Information Science*, 25(8):pages 1303–1322, 2011.

[82] Zhong-Xian Li, Xiao-Ming Yang, and Zongjin Li. Application of cement-based piezoelectric sensors for monitoring traffic flows. *Journal of transportation engineering*, 132(7):pages 565–573, 2006.

[83] Edward B Lieberman and Stephen Cohen. *New Technique for Evaluating Urban Traffic Energy Consumption and Emissions.* Number HS-020 305. 1976.

[84] Edward B Lieberman, Richard D Worrall, and JM Bruggeman. Logical design and demonstration of utcs-1 network simulation model. *Highway Research Record*, (409), 1972.

[85] Michael J Lighthill and Gerald Beresford Whitham. On kinematic waves. ii. a theory of traffic flow on long crowded roads. In *Proceedings of the Royal Society of London A: Mathematical, Physical and Engineering Sciences*, volume 229, pages 317–345. The Royal Society, 1955.

[86] Pao-Tsun Lin, Shun-Feng Su, and Tsu-Tian Lee. Support vector regression performance analysis and systematic parameter selection. In *Neural Networks, 2005. IJCNN'05. Proceedings. 2005 IEEE International Joint Conference on*, volume 2, pages 877–882. IEEE, 2005.

[87] Wei-Hua Lin, Qingying Lu, and Joy Dahlgren. Dynamic procedure for short-term prediction of traffic conditions. *Transportation Research Record: Journal of the Transportation Research Board*, (1783):pages 149–157, 2002.

[88] Jason Liu and Yue Li. Parallel hybrid network traffic models. *Simulation*, 85(4):pages 271–286, 2009.

[89] Junshui Ma, James Theiler, and Simon Perkins. Accurate on-line support vector regression. *Neural Computation*, 15(11):pages 2683–2703, 2003.

[90] Hani S Mahmassani. Dynamic traffic simulation and assignment: Models, algorithms and application to atis/atms evaluation and operation. In *Operations Research and Decision Aid Methodologies in Traffic and Transportation Management*, pages 104–135. Springer, 1998.

[91] Stephen Marsland. *Machine learning: an algorithmic perspective*. CRC Press, 2011. ISBN: 9781420067187.

[92] Kishan Mehrotra, Chilukuri K Mohan, and Sanjay Ranka. *Elements of artificial neural networks*. MIT press, 1997. ISBN: 9780262133289.

[93] JC Miles and Andrew J Walker. The potential application of artificial intelligence in transport. In *IEEE Proceedings-Intelligent Transport Systems*, volume 153, pages 183–198. IEEE, 2006.

[94] Luz Elena Y Mimbela and Lawrence A Klein. Summary of vehicle detection and surveillance technologies used in intelligent transportation systems. *Transportation Research Board*, page pages 211, 2000.

[95] Jae H Min and Young-Chan Lee. Bankruptcy prediction using support vector machine with optimal choice of kernel function parameters. *Expert systems with applications*, 28(4):pages 603–614, 2005.

[96] Juan M Morales and Jeffrey F Paniati. Two-lane traffic simulation: A field evaluation of roadsim. *Transportation Research Record*, (1100), 1986.

[97] Kai Nagel and Michael Schreckenberg. A cellular automaton model for freeway traffic. *Journal de physique I*, 2(12):pages 2221–2229, 1992.

[98] D Nikovski, N Nishiuma, Y Goto, and H Kumazawa. Univariate short-term prediction of road travel times. In *Intelligent Transportation Systems, 2005. Proceedings. 2005 IEEE*, pages 1074–1079. IEEE, 2005.

[99] Iwao Okutani and Yorgos J Stephanedes. Dynamic prediction of traffic volume through Kalman filtering theory. *Transportation Research Part B: Methodological*, 18(1):pages 1–11, 1984.

[100] A. Padiath, L. Vanajakshi, S.C. Subramanian, and H. Manda. Prediction of traffic density for congestion analysis under indian traffic conditions. In *12th International IEEE Conference on Intelligent Transportation Systems.*, pages 1–6, Oct 2009.

[101] Dongjoo Park and Laurence R Rilett. Forecasting freeway link travel times with a multilayer feedforward neural network. *Computer-Aided Civil and Infrastructure Engineering*, 14(5):pages 357–367, 1999.

[102] Stephen K. Park and Keith W. Miller. Random number generators: good ones are hard to find. *Communications of the ACM*, 31(10):pages 1192–1201, 1988.

[103] Harold J Payne. Models of freeway traffic and control. *Mathematical models of public systems*, 1971.

[104] Fengxiang Qiao, Hai Yang, and William HK Lam. Intelligent simulation and prediction of traffic flow dispersion. *Transportation Research Part B: Methodological*, 35(9):pages 843–863, 2001.

[105] Li Qu, Li Li, Yi Zhang, and Jianming Hu. Ppca-based missing data imputation for traffic flow volume: a systematical approach. *IEEE Transactions on Intelligent Transportation Systems*, 10(3):pages 512–522, 2009.

[106] R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2015.

[107] Sridhar Ramaswamy, Pablo Tamayo, Ryan Rifkin, Sayan Mukherjee, Chen-Hsiang Yeang, Michael Angelo, Christine Ladd, Michael Reich, Eva Latulippe, Jill P Mesirov, et al. Multiclass cancer diagnosis using tumor gene expression signatures. *Proceedings of the National Academy of Sciences*, 98(26):pages 15149–15154, 2001.

[108] Paul I Richards. Shock waves on the highway. *Operations research*, 4(1):pages 42–51, 1956.

[109] Henry Roncancio, André Carmona Hernandes, and Marcelo Becker. Vision-based system for pedestrian recognition using a tuned svm classifier. In *Engineering Applications (WEA), 2012 Workshop on*, pages 1–6. IEEE, 2012.

[110] Sam T Roweis and Lawrence K Saul. Nonlinear dimensionality reduction by locally linear embedding. *Science*, 290(5500):pages 2323–2326, 2000.

[111] Grzegorz Rozenberg, Thomas Bck, and Joost N Kok. *Handbook of natural computing*. Springer Publishing Company, Incorporated, 2011. 9783540929116.

[112] Rozvoj modernich dopravnich inteligentnich systemu. www.romodis.cz. `http://www.romodis.cz`.

[113] J David Schaffer. Multiple objective optimization with vector evaluated genetic algorithms. In *Proceedings of the 1st International Conference on Genetic Algorithms, Pittsburgh, PA, USA, July 1985*, pages 93–100, 1985.

[114] Jiuh-Biing Sheu. A stochastic modeling approach to dynamic prediction of section-wide inter-lane and intra-lane traffic variables using point detector data. *Transportation Research Part A: Policy and Practice*, 33(2):pages 79–100, 1999.

[115] Scott W Sibley. Netsim for microcomputers. *Public Roads*, 49(HS-039 378), 1985.

[116] Michaela Šikulová and Lukáš Sekanina. Coevolution in cartesian genetic programming. In *Genetic Programming*, pages 182–193. Springer, 2012.

[117] Brian L Smith and Michael J Demetsky. Short-term traffic flow prediction: neural network approach. *Transportation Research Record*, (1453), 1994.

[118] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression. *Statistics and computing*, 14(3):pages 199–222, 2004.

[119] Anthony Stathopoulos and Matthew G Karlaftis. A multivariate state space approach for urban traffic flow modeling and prediction. *Transportation Research Part C: Emerging Technologies*, 11(2):pages 121–135, 2003.

[120] Antony Stathopoulos, Loukas Dimitriou, and Theodore Tsekeris. Fuzzy modeling approach for combined forecasting of urban traffic flow. *Computer-Aided Civil and Infrastructure Engineering*, 23(7):pages 521–535, 2008.

[121] James V Stone. *Independent component analysis*. Wiley Online Library, 2004. ISBN: 9780262693158.

[122] Shiliang Sun, Rongqing Huang, and Ya Gao. Network-scale traffic modeling and forecasting with graphical lasso and neural networks. *Journal of Transportation Engineering*, 2012.

[123] WY Szeto, Bidisha Ghosh, Biswajit Basu, and Margaret O'Mahony. Multivariate traffic forecasting technique using cell transmission model and sarima model. *Journal of Transportation Engineering*, 135(9):pages 658–667, 2009.

[124] S. Takaba, T. Morita, T. Hada, T. Usami, and M. Yamaguchi. Estimation and measurement of travel time by vehicle detectors and license plate readers. In *Vehicle Navigation and Information Systems Conference, 1991*, volume 2, pages pages 257–267, Oct 1991.

[125] Man-Chun Tan, SC Wong, Jian-Min Xu, Zhan-Rong Guan, and Peng Zhang. An aggregation approach to short-term traffic flow prediction. *IEEE Transactions on Intelligent Transportation Systems*, 10(1):pages 60–69, 2009.

[126] Francis EH Tay and Lijuan Cao. Application of support vector machines in financial time series forecasting. *Omega*, 29(4):pages 309–317, 2001.

[127] Joshua B Tenenbaum, Vin De Silva, and John C Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):pages 2319–2323, 2000.

[128] Luis Torgo. *Data mining with R: learning with case studies*. Chapman & Hall/CRC, 2010. 9781439810187.

[129] Martin Treiber and Arne Kesting. Traffic flow dynamics. *Traffic Flow Dynamics: Data, Models and Simulation, Springer-Verlag Berlin Heidelberg*, 2013.

[130] RJ Troutbeck. Overtaking behaviour on narrow two-lane two way rural roads. *Australian Road Research*, 12(5), 1984.

[131] Olga Troyanskaya, Michael Cantor, Gavin Sherlock, Pat Brown, Trevor Hastie, Robert Tibshirani, David Botstein, and Russ B Altman. Missing value estimation methods for dna microarrays. *Bioinformatics*, 17(6):pages 520–525, 2001.

[132] Belle L Tseng, Ching-Yung Lin, and John R Smith. Real-time video surveillance for traffic monitoring using virtual line analysis. In *Multimedia and Expo, 2002. ICME'02. Proceedings. 2002 IEEE International Conference on*, volume 2, pages 541–544. IEEE, 2002.

[133] Ambros Valach, Tecl and Vyskocilova. Vyse ztrat z dopravni nehodovosti na pozemnich komunikacich za rok 2013. `http://www.czrso.cz/clanky/vyse-ztrat-z-dopravni-nehodovosti-na-pozemnich-komunikacich-za-rok-2013/`. Accessed: 2015-10-20.

[134] M Van Aerde. Modelling of traffic flows, assignment and queueing in integrated freeway/traffic signal networks. *Transportation Research Record*, pages pages 306–315, 1985.

[135] CP Van Hinsbergen, JW Van Lint, and FM Sanders. Short term traffic prediction models. In *PROCEEDINGS OF THE 14TH WORLD CONGRESS ON INTELLIGENT TRANSPORT SYSTEMS (ITS), HELD BEIJING, OCTOBER 2007*, 2007.

[136] Eleni I Vlahogianni, John C Golias, and Matthew G Karlaftis. Short-term traffic forecasting: Overview of objectives and methods. *Transport reviews*, 24(5):pages 533–557, 2004.

[137] Eleni I Vlahogianni, Matthew G Karlaftis, and John C Golias. Optimized and meta-optimized neural networks for short-term traffic flow prediction: a genetic approach. *Transportation Research Part C: Emerging Technologies*, 13(3):pages 211–234, 2005.

[138] Eleni I Vlahogianni, Matthew G Karlaftis, and John C Golias. Short-term traffic forecasting: Where we are and where we're going. *Transportation Research Part C: Emerging Technologies*, 43:pages 3–19, 2014.

[139] Wendy Weijermars and Eric Van Berkum. Analyzing highway flow patterns using cluster analysis. In *Proceedings of Intelligent Transportation Systems*, pages 308–313. IEEE, 2005.

[140] Chih-Hung Wu, Gwo-Hshiung Tzeng, Yeong-Jia Goo, and Wen-Chang Fang. A real-valued genetic algorithm to optimize the parameters of support vector machine for predicting bankruptcy. *Expert systems with applications*, 32(2):pages 397–408, 2007.

[141] Chih-Hung Wu, Gwo-Hshiung Tzeng, and Rong-Ho Lin. A novel hybrid genetic algorithm for kernel function and parameter optimization in support vector regression. *Expert Systems with Applications*, 36(3):pages 4725–4735, 2009.

[142] Chun-Hsin Wu, Jan-Ming Ho, and Der-Tsai Lee. Travel-time prediction with support vector regression. *Intelligent Transportation Systems, IEEE Transactions on*, 5(4):pages 276–281, 2004.

[143] Chun-Hsin Wu, Jan-Ming Ho, and D.T. Lee. Travel-time prediction with support vector regression. *IEEE Transactions on Intelligent Transportation Systems,*, 5(4):pages 276–281, Dec 2004.

[144] Chun-Hsin Wu, Da-Chun Su, Justin Chang, Chia-Chen Wei, Jan-Ming Ho, Kwei-Jay Lin, and DT Lee. An advanced traveler information system with emerging network technologies. In *Proc. 6th Asia-Pacific Conf. Intelligent Transportation Systems Forum*, pages 230–231, 2003.

[145] Chun-Hsin Wu, Da-Chun Su, Justin Chang, Chia-Chen Wei, Kwei-Jay Lin, and Jan-Ming Ho. The design and implementation of intelligent transportation web services. In *IEEE International Conference on E-Commerce*, pages 49–52. IEEE, 2003.

[146] Yuanchang Xie, Yunlong Zhang, and Zhirui Ye. Short-term traffic volume forecasting using kalman filter with discrete wavelet decomposition. *Computer-Aided Civil and Infrastructure Engineering*, 22(5):pages 326–334, 2007.

[147] Cleber Zanchettin, Byron Leite Dantas Bezerra, and Washington W Azevedo. A knn-svm hybrid model for cursive handwriting recognition. In *The 2012 International Joint Conference on Neural Networks (IJCNN),*, pages 1–8. IEEE, 2012.

[148] Changshui Zhang, Shiliang Sun, and Guoqiang Yu. A bayesian network approach to time series forecasting of short-term traffic flows. In *International IEEE Conference on Intelligent Transportation Systems*, pages 216–221. IEEE, 2004.

[149] Weizhong Zheng, Der-Horng Lee, and Qixin Shi. Short-term freeway traffic flow prediction: Bayesian combined neural network approach. *Journal of transportation engineering*, 132(2):pages 114–121, 2006.

[150] Eckart Zitzler and Lothar Thiele. Multiobjective evolutionary algorithms: a comparative case study and the strength pareto approach. *IEEE transactions on Evolutionary computation*, 3(4):pages 257–271, 1999.