Technical University of Crete

Adaptive Fine-Tuning for Large-Scale Nonlinear Traffic Control Systems

Thesis submitted in partial fulfillment of
the requirements for the degree of
Doctor of Philosophy

by
Anastasios Kouvelas

Chania, Greece, January 2011

tasos@dssl.tuc.gr

Chania, Greece, January 2011

The thesis is approved by the following jury:

Markos Papageorgiou (Supervisor) .....
Professor, Department of Production and Management Engineering
Technical University of Crete

Elias Kosmatopoulos (Member of committee in charge) .....
Associate Professor, Department of Electrical and Computer Engineering
Democritus University of Thrace

Dimitrios Rovas (Member of committee in charge) .....
Assistant Professor, Department of Production and Management Engineering
Technical University of Crete

Vassilis Kouikoglou .....
Professor, Department of Production and Management Engineering
Technical University of Crete

Georgios Stavroulakis .....
Professor, Department of Production and Management Engineering
Technical University of Crete

Giorgos Stavrakakis .....
Professor, Department of Electronic and Computer Engineering
Technical University of Crete

Iasson Karafyllis .....
Assistant Professor, Department of Environmental Engineering
Technical University of Crete

Acknowledgements

With this thesis, a journey of almost 11 years of studies at the Technical University of Crete comes to an end. During these years there were some bad times but a lot of good times as well. After all, the journey itself is always more important than the final destination. At this point I would like to acknowledge all the people who helped and supported me throughout my studies.

Foremost, I would like to thank my supervisors, Professor Markos Papageorgiou and Professor Elias Kosmatopoulos, for their advice and guidance. Their encouragement, interest, enthusiasm and wisdom made the accomplishment of this thesis possible. I have learnt a lot from both of them in many fronts, extending well beyond academics.

I would also like to thank Professor Dimitrios Rovas for his thoughtful critique and comments as a member of the committee in charge, as well as the rest members of the doctoral committee, Prof. Vassilis Kouikoglou, Prof. Georgios Stavroulakis, Prof. Giorgos Stavrakakis, and Prof. Iasson Karafyllis, for their valuable feedback and the stimulating discussions.

I am also grateful to my fellow researchers at the Dynamic Systems & Simulation Laboratory. All the members of the lab, as well as my fellow students, have been both colleagues and friends. We had had a great time at the lab working, but also having fun. Above all, Dr. Konstantinos Aboudolas has been a great office-mate and a good friend.

I would also like to thank my parents, Giannis and Magdalini, and my two brothers, Marios and Nikos, for their endless support all these years. Without their constant encouragement it wouldn't be possible for me to pursue my goals.

Finally, I want to thank all those that stood by me these four years. These include members of my family, close friends and colleagues in Technical University of Crete. Thank you!

Anastasios Kouvelas

January 2011

Short Biography

Anastasios Kouvelas was born in Athens, Greece, in 1981. He received the Diploma degree from the Department of Production & Management Engineering, Technical University of Crete, Greece, in 2004 (2^{nd} in his class), where he continued his studies at the Operational Research postgraduate program. He received the M.Sc. degree in 2006 and since 2007 he is a Ph.D. student at the same Department.

From 2003 he is a Research and Teaching Assistant at the Dynamic Systems & Simulation Laboratory, Technical University of Crete. He has been teaching as an assistant several undergraduate and postgraduate courses. In 2009 he was a Short Term Visiting Scholar at the Center for Advanced Transportation Technologies (CATT) of the Electrical Engineering Department, University of Southern California, Los Angeles. His main research interests include modeling, simulation and control of urban and motorway traffic networks, learning techniques, neural networks, stochastic approximation, adaptive optimisation and intelligent transportation systems.

Mr. Kouvelas is the co-author of some 15 journal and conference papers and he has participated in various research projects funded by the European Union. He is a member of the Technical Chamber of Greece.

Abstract

Despite the continuous advances in the fields of control and computing, the design and deployment of an efficient Large-scale Nonlinear Traffic Control System (LNTCS) remains a significant objective. This is mainly due to the complexity and the strong nonlinearities involved in the modeling of traffic flow processes. Practical control design approaches are often based on simplified models about the system dynamics, leading to LNTCS with suboptimal performance, as the use of more complex models of effective LNTCS is virtually unavoidable in most complex control system applications.

The ultimate performance of a designed or operational LNTCS (e.g. urban signal control, or ramp metering) depends on two main factors: (a) the exogenous influences, e.g. demand, weather conditions, incidents, and (b) the values of some design parameters included within the LNTCS. When a new control algorithm is implemented there is a period of, sometimes tedious, fine-tuning activity that is needed in order to elevate the control algorithm to its best achievable performance. Fine-tuning concerns the selection of appropriate (or even optimal) values for a number of design parameters included in the control strategy.

Moreover, the continuous medium- and long-term variations of the traffic system dynamics call for a frequent or even continuous maintenance of LNTCSs. When an operational but "aged" control algorithm needs to be updated the same fine-tuning procedure has to take place, which – if done properly – is extremely costly. Typically, this fine-tuning procedure is conducted manually, via trial-and-error, relying on expertise and human judgment and without the use of a systematic approach. Currently, a considerable amount of human effort and time is spent for initialization or calibration of operational LNTCSs, which does not always lead to a desirable outcome. In many cases, the result is that system maintenance is neglected and the system performance deteriorates year after year.

This thesis introduces and analyzes a new learning/adaptive algorithm that enables automatic fine-tuning of LNTCS, so as to reach the maximum performance that is achievable with the utilized control strategy. The proposed Adaptive Fine Tuning (AFT) algorithm is aiming at replacing the conventional manual optimization practise with a fully automated online procedure. The thesis provides a detailed analysis of the algorithm as well as a step-by-step application description. Finally, application results of the algorithm to real-time fine-tuning problems of general LNTCS are presented.

The efficiency and online feasibility of AFT algorithm is investigated through extensive simulation experiments for two LNTCS. The first test case is a large-scale ramp metering control problem. A multivariable ramp metering regulator is applied to the stretch of the

Monash motorway in Melbourne, Australia. The latter test case corresponds to the application of an urban signal control strategy to the road network of Chania, in Greece . In both simulated cases, AFT is used in order to iteratively fine-tune the design parameters of the system. The simulation results illustrate the algorithm's efficiency and real-time applicability. AFT is seen to provide efficient automatic fine-tuning of the design parameters of general LNTCS, guaranteeing safe and convergent behavior.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

This first chapter introduces the reader to the problem under study. Section 1.1 presents the motivation of this work and Section 1.2 the thesis goals. In Section 1.3 an outline of the document is provided. Finally, in Section 1.4 publications related to the research of this thesis are presented.

1.1 Motivation

Despite the continuous advances in the fields of control and computing, the design and deployment of an efficient Large-scale Nonlinear Traffic Control System (LNTCS) remains a significant objective, mainly because of the involved complexity and the strong nonlinearities. The ultimate performance of a designed or operational LNTCS (e.g. urban signal control, or ramp metering, or Variable Speed Limit (VSL) control) depends on two main factors: (a) the exogenous influences, e.g. demand, weather conditions, incidents, and (b) the values of some design parameters included in the LNTCS.

As a matter of fact, when a new control algorithm is implemented (or an operational but "aged" control algorithm needs to be updated), there is a period of, sometimes tedious, fine-tuning activity that is needed in order to elevate the control algorithm to its best achievable performance. Fine-tuning concerns the selection of appropriate (or even optimal) values for a number of design parameters included in the control strategy. Typically, this fine-tuning procedure is conducted manually, via trial-and-error, relying on expertise and human judgment and without the use of a systematic approach.

Currently, a considerable amount of human effort and time is spent for calibration of operational LNTCSs. Minor changes in the transport system infrastructure (e.g. installing a new Variable Message Sign (VMS) in a motorway network, modifying the traffic light signal phasing at an urban junction, deploying a new bus in a public transport system or

a new Automated Guided Vehicle (AGV) in a seaport container terminal) may require the involvement of significant human effort and time in order to re-adjust and re-program the LNTCS decision making mechanisms.

Moreover, the continuous medium- and long-term variations of the overall transport system dynamics (e.g. due to changes of traffic demand or number of passengers using the particular transport system) call for a frequent or even continuous maintenance of LNTCSs, which – if done properly – it is extremely costly. In many cases, the result is that system maintenance is neglected and the system performance deteriorates year after year.

1.2 Thesis objectives

Urban and motorway traffic control systems, LNTCSs for public transport and LNTCSs for large-scale railway, airport and seaport operations are all specific examples of LNTCSs that call for calibration while the system is in operation. In all these systems, the maintenance procedure involves the re-calibration, re-adjustment and re-programming of hundreds of parameters, rules, operational schedules, decision-making mechanisms, etc., which influence the transport system operations in a highly complex manner. Moreover, the use of heuristic, trial-and-error, experience-based techniques, while the system is in operation, involves the risk of poor system performance over a lengthy period of time. This, may lead to poor quality service problems, delays, severe congestion and increased Green House Gas (GHG) emissions during this period. It is finally worth noting, that the involvement of the human factor for the installation, maintenance and renovation of LNTCSs also involves the risk of unsafe operations. Human mistakes due to lack of expertise, exhaustive working conditions, etc., may lead to decisions/actions that put safety at stake.

In general, the same process that is required for calibration is also used in the initial fine-tuning of the control system, during its first installation. Both tasks (initial fine-tuning and calibration), are performed (if at all) by experienced personnel in the lack of an automated approach. Thus, there is no guarantee that the overall fine-tuning and/or maintenance procedure will end up successfully. In some cases, the LNTCS has never achieved a satisfactory performance in the first place. Such an example, is the reported case of the urban management strategy SCOOT (the most popular urban signal management strategy worldwide) in the city of Nijmegen, in The Netherlands [54], where the SCOOT application was abandoned completely in the end.

The main contributions of this research include:

- Development and presentation of AFT methodology, which allows for automated adaptive fine-tuning of LNTCSs. This methodology aims at replacing the conventional

manual optimization practise with a fully automated online procedure.

- Mathematical analysis of the proposed algorithm and presentation of its connection to other famous stochastic approximation methodologies.
- Thorough investigation (via simulation experiments) of the algorithm's efficiency and feasibility under different problems and scenarios.
- Provision of general guidelines about the application of AFT to general large-scale fine-tuning problems.

1.3 Document outline

The thesis is organized in a series of self-contained chapters. This first introductory chapter presents the motivation and the thesis goals. The outline of the rest of the document is as following:

- Chapter 2 presents a short overview of automatic control methods. Optimization and control techniques that provide optimal control strategies are discussed, as well as the open-loop and closed-loop optimal control regulators. Finally, the need for fine-tuning in large-scale, complex control systems applications is described.
- Chapter 3 presents the state of the art in parameter estimation/optimization methodologies. It provides an analytical presentation of Stochastic Approximation (SA) and analyzes the general principles of designing SA search algorithms. Also, the popular SA algorithms FDSA, RDSA and SPSA are thoroughly described.
- Following, Chapter 4 explores the AFT algorithm. Moreover, it presents a comparison of the use of theoretical/simulation-based methods and adaptive and neural/learning methods as a solution to the fine-tuning problem. The concept of universal approximators is also discussed. Finally, the chapter presents efficient techniques about calculating stepsizes for SA methods.
- Chapter 5 presents the application of AFT algorithm to a large-scale ramp metering problem. AFT is applied to the Monash motorway in Melbourne, Australia and the macroscopic simulator METANET is used for the simulation experiments. This chapter examines and analyzes in details the results of the simulation experiments.
- Chapter 6 presents the application of AFT algorithm to a large-scale urban signal control problem. AFT is applied for fine-tuning to the urban road network of the city

of Chania, in Greece and the microscopic simulator AIMSUN is used for the simulation experiments. This chapter examines and analyzes in details the results of the simulation experiments.

- The thesis is concluded in Chapter 7, which summarizes its findings and results. Finally, future perspectives are presented, which can help to the extension of these results.

1.4 Publications related to thesis

The findings of this thesis have contributed to the following publications:

Journals

1. E. Kosmatopoulos and A. Kouvelas. Large-Scale Nonlinear Control System Fine-Tuning through Learning. IEEE Transactions on Neural Networks, vol. 20, no. 6, pp. 1009–1023, 2009.

2. A. Kouvelas, K. Aboudolas, E. Kosmatopoulos, M. Papageorgiou. Adaptive Performance Optimization for Large-Scale Traffic Control Systems. Under review in IEEE Transactions on Intelligent Transportation Systems.

Conferences

1. E. Kosmatopoulos, M. Papageorgiou, Y. Wang, I. Papamichail, A. Kouvelas. AFT2: An Automated Maintenance and Calibration Tool for Traffic Management & Control Systems. In proceedings of the 11th International IEEE Conference on Intelligent Transportation Systems, pp. 97–104, Beijing, China, 12–15 October 2008.

2. A. Kouvelas, E. Kosmatopoulos, M. Papageorgiou, K. Aboudolas. Adaptive Performance Optimization for Large-Scale Traffic Control Systems. In proceedings of the 12th IFAC Symposium on Control in Transportation Systems, pp. 76–83, Redondo Beach, California, USA, 2–4 September 2009.

Workshop

1. A. Kouvelas, E. Kosmatopoulos, M. Papageorgiou, K. Aboudolas. Adaptive Optimization with Satisfactory Transient Performance for Large-Scale Traffic Control Systems. 2nd NEARCTIS Workshop, University College London, UK, 13 November 2009.

Chapter 2

Automatic control systems

This chapter presents a short overview of the existing automatic control methods. Section 2.1 provides a brief introduction to Automatic Control Systems and Sections 2.2 and 2.3 describe some general guidelines about the procedure of designing a control strategy. Section 2.4 summarizes the optimization methodologies used for solving optimal control problems and Section 2.5 discusses the open-loop and closed-loop optimal control regulators. In Sections 2.6 all the necessary procedures for deploying a control system application are presented. Finally, Section 2.7 describes the need for fine-tuning in large-scale, complex control systems and Section 2.8 remarks the two traffic control systems considered in this thesis.

2.1 Introduction

Automatic Control comprises those theoretical methods and practical procedures that enable the development of technical systems capable of accomplishing autonomously certain pre-specified tasks. Figure 2.1 illustrates the basic elements of an automatic control system. The **process** (e.g. traffic flow in an urban network) includes all technical or physical phenomena that should be influenced according to specific goals. The dynamic process changes depend upon:

- Some external quantities that are assumed independent of the dynamic evolution of the process (e.g., in the case of urban traffic, the external quantities may be the traffic lights, the traffic demand, the origin-destination pattern, the incidents, the environmental conditions, etc.)
- The proper behavior of the process according to its technical and/or physical nature (e.g. the travel times or the queue storage of vehicles on an urban link, the flow or storage capacity, etc.).

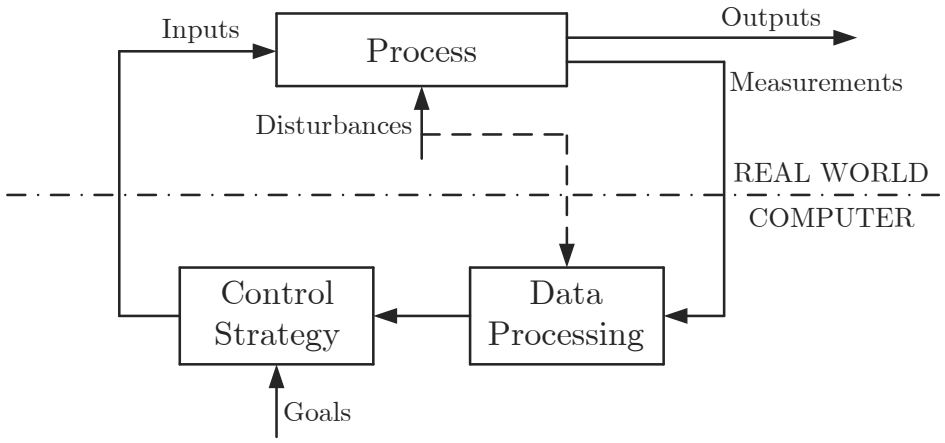

Figure 2.1: Basic elements of an automatic control system.

The **external quantities** may be classified in:

- **Inputs**, whose values may be selected from an admissible control region (e.g. the traffic lights, the variable message signs, etc.).
- **Disturbances**, whose values cannot be manipulated but may possibly be directly measurable via appropriate devices (e.g. traffic demand), or may be estimated or predicted via appropriate algorithms (e.g. traffic demand, origin-destination pattern, etc.).

The process **outputs** are the quantities chosen to represent the behavioral aspects of interest (e.g. the outputs of urban traffic may be the total travel time, the queue lengths, etc.). The **data processing** block in Figure 2.1 comprises the estimation and/or prediction tasks, based on real-time measurements of internal process quantities or disturbances.

The task of the **control strategy** is to specify in real time the process inputs, based on available measurements/estimations/predictions, so as to achieve the pre-specified goals regarding the process outputs, despite the influence of various disturbances.

If this task is undertaken by a human operator, we have a manual control system. On the other hand, in an automatic control system, this task is undertaken by an algorithm (the control strategy). In modern and/or complex systems, the control strategy as well as the data processing algorithms are typically implemented in a computer.

It is not too difficult to analyze any automatic control system within the frame of Figure 2.1, even if the corresponding processes may be of very different nature, such as robots, airplanes, chemical processes, water or gas networks, etc. The specification of inputs and

measurements for a particular automatic control system is closely related to technological issues. On the other hand, the data processing and particularly the control strategy blocks contain the system's "intelligence", i.e. its capability to face automatically and efficiently any situation arising due to the impact of the disturbances.

At this point it is necessary to make a distinction between the notions of **control strategy** and **mathematical model**. A mathematical model of a process comprises a number of equations that describe, with a more or less limited accuracy, the proper (internal) process behavior in the considered context. Hence, a mathematical model, fed with input and disturbance values, may be employed to calculate (with a certain accuracy) the corresponding output values or other internal quantities. For example, an urban traffic flow model, fed with the values of the traffic lights, the traffic demand, the origin-destination pattern, etc., may be used to calculate the corresponding queue lengths and travel times in all network links. **Dynamic** models describe the time development of the process phenomena. For one and the same process, there may be several useful models with different levels of resolution, accuracy and complexity. For example, for the urban traffic flow, existing models are microscopic or macroscopic, static or dynamic, stochastic or deterministic, etc.

On the other hand, a control strategy is an algorithm that makes the decisions regarding the control actions that should be applied at each instant in time, i.e. it calculates the input values. As we will see later, a control strategy may be designed on the basis of a mathematical model, or may even explicitly include a mathematical model, as a mean of assessing in real time the efficiency of this or that control action. Despite these connections, it appears important to distinguish clearly between the tool that imitates the process behavior (mathematical model) and the tool that makes control decisions (control strategy).

2.2 Designing a control strategy

This section discusses the issue of how should one design a control strategy. One possibility could be to try to imitate (to model) the behavior of a (real or hypothetical) human operator (expert system approach). Another possibility is to attempt to understand (to model) the process behavior and then apply systematic methods (automatic control approach) that lead to an adequate control strategy. The next sections of this chapter provide some basic information regarding the automatic control approach.

Automatic Control exists as an independent discipline since some 70 years. During this period in time, automatic control engineers have developed and refined a number of methods for the systematic design of efficient, reliable and robust control strategies, and they have applied these methods to a high number of processes (space, defense, robotics, chemical

processes, traffic, environment, etc.). The basic philosophy and the importance of these methods are related to their general applicability: they are not particular heuristics valid just for a specific process, but general methods applicable to any process that can be described by certain types of mathematical models, regardless the physical process nature (robot, airplane, traffic, environment, etc.). This general approach reaches its limits, if, for a specific process:

- There is a lack of understanding, i.e. no adequate model available.
- Certain complexity limits are exceeded.
- The process behavior is of a discrete or event-oriented or combinatorial nature.

Under these conditions, that have become more and more frequent in recent years, continuing efforts for developing general, efficient, and systematic methods do not always reach the maturity required for successful practical applications.

Before proceeding into more details regarding the automatic control approach, it is useful to present some features of the basic structure of an automatic control system as represented in Figure 2.1. The control system is characterized by a **closed-loop** structure, whereby the calculation of inputs is effectuated on the basis of measurements of process internal quantities, which, by their turn, are influenced by the inputs. Albeit, one could use an alternative structure. Assume availability of a process model of the type

$$y = f(u, d) \tag{2.1}$$

where the vectors y, u, and d include the outputs, the inputs, and the disturbances, respectively, while f is a generalized operator, e.g. a number of differential equations. Assume also availability of desired output values y_d. If (2.1) is invertible, one obtains

$$u_d = f^{-1}(d, y_d) \tag{2.2}$$

which corresponds to an **open-loop** control strategy (Figure 2.2) that makes no use of process measurements. The advantage of this structure, compared to the closed-loop, is the rapidity of control action: the strategy reacts immediately to disturbance variations without waiting for the disturbance impact to become visible in the internal process variables. Moreover, if the process itself is stable, there is no risk of obtaining an unstable system as in the closed-loop case.

Unfortunately, the disadvantages of open-loop control are much more important than the advantages. In fact, if no real process measurements are utilized, like in the open-loop structure, the real process state is never known, i.e. there is no way for the control strategy to know whether the real outputs y are actually close to y_d (the desired ones). Hence, the

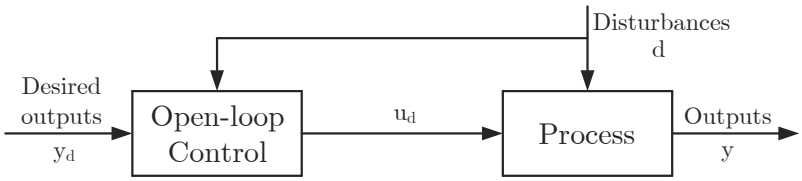

Figure 2.2: Open-loop control system.

control strategy will not react if y is far from y_d. This uncertainty about the real process state the may originate from: (a) the limited accuracy of any mathematical model; (b) the presence of non-measurable disturbances.

Because of this inherent uncertainty, any automatic control system is forced to include a closed-loop structure. Nevertheless, if measurements/estimates/predictions of some major disturbances are available, they may be used to ameliorate the control system efficiency as indicated in the dashed signal line in Figure 2.1.

2.3 The regulation problem

The regulation problem is a special case of the control system of Figure 2.1, whereby the control goal is to lead and maintain the process output y near pre-specified corresponding desired values y_d that are called **set values**. Moreover, it is usually assumed that the real outputs y are measurable in real-time. Regulation problems call for a **regulator**, i.e. a formula

$$u = R(y, d) \tag{2.3}$$

that guarantees $y \approx y_d$ despite the presence of disturbances. The major performance criterions for a regulator are:

- **Stability** (above any other consideration).
- **Rapidity** of response in case of a change of the set values or the disturbances.
- **Stationary accuracy**, i.e. $y \approx y_d$ under stationary conditions.
- **Robustness**, i.e. preservation of the control performance even if the real process behavior is not identical to the mathematical model used for the regulator design.

Automatic Control theory offers a number of methods and theoretical results for designing a regulator in a systematic and efficient way. A necessary condition for application of the Automatic Control theory to a particular process control problem is the availability of a mathematical model capable of describing the basic process behavior. In fact, the model to

be used for regulator design (the design model) may be quite simple, if it includes the major aspects of the process behavior and if the designed regulator is sufficiently robust. Most regulators resulting from application of Automatic Control methods are very simple, as they consist of one single equation (in the form of (2.3)), but their efficiency and reliability are usually much higher than those of human regulators. It is important to note, that when designing a regulator, the mathematical process model is only used off-line, i.e. the online application of (2.3) does not include any model equations.

Most Automatic Control methods are applicable to linear models, while the nonlinear control theory is less developed. In many cases, it is possible to linearize a nonlinear model (e.g. around the set values) before the regulator design, which may call for special measures during control operations in order to avoid practical difficulties.

The regulator design for SISO (Single-Input-Single-Output) processes is relatively simple (though not trivial). For example, ALINEA (see Chapter 5) is a SISO regulator. The methods used for linear SISO cases are simple and usually require basic knowledge of Automatic Control theory. Regarding MIMO (Multiple-Input-Multiple-Output) processes, the regulator design and the corresponding methods become more elaborated. In both cases, SISO and MIMO, a good knowledge of the methodology and a certain experience of the designer are essential, in order to come up with efficient control strategies. The split control module of TUC strategy (see Chapter 6) is an example of a MIMO regulator. The design methods for linear MIMO regulators are more difficult and advanced. Such methods are the Linear-Quadratic (LQ) optimization, pole assignment methods, decentralized control, hierarchical control, etc. Particular attention should be paid to the robustness properties of the designed regulators, via recently developed powerful methods and tools.

Further methods for particular classes of regulators are available within Automatic Control theory, like, for example, **nonlinear regulators** (for nonlinear processes) and **adaptive regulators**, whereby the regulator parameters are adjusted automatically in real-time by suitable mechanisms, in order to account for process uncertainties or for time-varying process behavior.

2.4 Optimization theory

Optimal control problems may be considered as a particular area of the broader Optimization Theory. The basic problem in optimization theory is to specify, within a given space, an optimal solution that minimizes a criterion value $J \in R$ subject to pre-specified constraints. According to the nature of the searched space and the constraints, one may distinguish different classes of optimization problems with corresponding solution methodologies, as for

example:

- Static optimization (gradient methods are used that lead to local or global minima).
- Dynamic optimization (using variational calculus or **dynamic programming**).
- Combinatorial optimization (large-scale problems face the so called **combinatorial explosion** – NP hard).
- Stochastic optimization (the optimization goal is to minimize the **expected value** of a criterion $J \in R$, see Chapter 3).
- Game theory (including two or more (instead of one) decision makers with competing objective criteria).
- Multicriteria optimization (minimization of more than one (typically conflicting) objective criteria).
- Heuristics (including general structural heuristics, surveillance and emergency heuristics, specifications heuristics).

For some problem types, the solution may be calculated analytically, which is an advantage for practical implementation of the results. In case of an algorithmical solution by use of a computer, the issue of the required computational effort becomes important, particularly for real-time applications. In fact, the application of optimization methodologies typically hits its limits due to the complexity of the problem and the scalability of the available optimization methodologies.

In conclusion, one may state that, despite the variety of theoretical results and algorithms offered by Automatic Control and Optimization, it seems almost inevitable to complement a rigorously designed control strategy by various heuristics, aiming to address particular practical aspects. For simple systems, the development of these heuristics may be relatively easy. But for complex systems, including dozens of control inputs, sub-loops, and measurements, it is desirable to tackle this problem in a more systematic, theoretically founded way. Some concepts and tools in this direction are provided by a special branch within Automatic Control theory, namely the theory of **Discrete Event Dynamic Systems**.

2.5 Optimal control strategies

This section addresses the general case of the control system of Figure 2.1 under the assumption that the control goal can be expressed as the minimization (or maximization) of

a quantity J (the **objective function** or optimization **criterion**). This quantity may, for example, correspond to a system output y that depends on the inputs u and the internal process variables x

$$\min J(u, x) = y(u, x). \tag{2.4}$$

In the case of urban traffic, this criterion typically corresponds to the minimization of total travel time in the considered network.

The internal variables x depend upon the inputs u and disturbances d according to a mathematical model. A great part of dynamic processes may be described by a **state-space model**, that has the general form

$$\dot{x} = f(x, u, d) \tag{2.5}$$

where the vector x comprises the **state variables**. Note that if the **initial condition**, i.e. the value of x for time $t = 0$, is known

$$x(0) = x_0 \tag{2.6}$$

and the time trajectories $u(t)$, $d(t)$, $t \in [0, T]$, are also given, the differential equations (2.5) may be resolved to deliver the corresponding state trajectory $x(t)$ over the same time period $[0, T]$.

The choice of inputs u is usually limited due to physical or technical constraints that define an **admissible control region** via a set of inequalities

$$h(x, u, d, t) \leq 0, \forall t \in [0, T]. \tag{2.7}$$

In order to obtain a **closed-loop solution**, that also considers available disturbance predictions, one should consider the following optimization problem:

Given the disturbance trajectories $d(t)$, $t \in [0, T]$, find a function R

$$u(t) = R[x(t), t], t \in [0, T] \tag{2.8}$$

that minimizes the criterion J subject to the model equations (2.5) and the constraints (2.7).

Note that the solution of this optimization problem derives a **function** $R(x, t)$, called the **control law**, that may be executed in real-time by use of state measurements x (closed-loop solution). This solution, is independent of the initial condition and hence applicable anywhere in the space (x, t). Albeit, an analytical solution is only feasible for problems of simple structure (e.g. linear model, quadratic criterion, no constraints) or low dimensions. In

this case, the function (2.8) will be delivered analytically (e.g. $u(t) = -x(t)$). But for large-scale and complex problems, the analytical solution is not always straightforward and they call for a numerical solution. On the other hand, the computational effort for a numerical solution (using dynamic programming) increases exponentially with this problem dimension, something that limits applicability of the procedure to relatively low order problems.

This discussion, of the available optimal control methodology, obviously leads to a dilemma when considering high dimensional control problems. On the one hand, there is the possibility of an open-loop control structure with its corresponding important drawbacks. If the model (2.5) or the disturbance predictions are not accurate, the real state variables and the real process outputs will be accordingly different from the optimization results. Moreover, it should be noted, that the control output derived by an open-loop regulator, is only optimal for the particular initial condition considered in (2.6). On the other hand, the generation of a closed-loop control structure becomes computationally intractable for large-scale control problems. In order to avoid this dilemma and obtain efficient and feasible solutions, some suboptimal procedures have been developed for practical applications:

1. Hierarchical multilayer control.

2. Repetitive optimization (with rolling horizon).

3. Combination of methods (1) and (2).

At this point, it should also be noted, that all the procedures described above are also applicable to **discrete-time optimal control problems**, which are based on a discrete-time dynamic model of the type

$$x(t+1) = f\left[x(t), u(t), d(t)\right] \tag{2.9}$$

instead of (2.5), where t denotes the discrete-time index. This thesis deals with two discrete-time traffic control problems that are analytically described in the following chapters.

In conclusion, optimal control theory allows for the direct development of control strategies only for problems of simple structure or low dimension. In more complex application cases, it is necessary to embed the optimal problem solution in a (possibly hierarchical) control structure, so as to circumvent the accumulation of unavoidable uncertainties. However, the quality of decisions delivered via the solution of an optimal control problem is clearly superior to a human operator's performance. This superiority becomes even more pronounced for complex problems. In some cases, the decisions delivered by the optimal control problem solution might even appear strange or inexplicable at first view, calling for a more careful analysis of the results, so as to understand and appreciate their sensible background and

express them in human reasoning terms. The origin of this "intelligence", is a general theory that does not rely on empirical and questionable heuristic rules; rather, it allows for an exhaustive but efficient search in the space of all feasible decisions and the selection of the best (optimal) ones.

2.6 Automatic control applications procedures

As already mentioned, Automatic Control became an independent scientific theory only some 70 years ago. Nevertheless, control problems were encountered even before 1940 in several technical domains. The oldest regulators (integrated in water distribution works) appeared in ancient Egypt and Greece, but it was only after the Watt vapor engine that regulators could be physically distinguished from the process under control. It was only in this century, that the concept and the effect of **feedback loops** was explicitly studied (e.g. in the context of electrical circuits), largely understood, and intentionally employed for system regulation.

Until 1940, regulation problems and their respective solutions remained at a low scientific level and addressed only particular application needs. The engineers in charge of water works, electrical, chemical, and mechanical systems invented independently regulation mechanisms based on feedback loops that stabilized, mostly without theoretical justification, the corresponding quantities, without really understanding how and why these control systems worked. It is only after 1940, that one begins to realize that the behavior of processes of completely different nature may be very similar, once expressed in mathematical equations. As a consequence, the development of regulation methodologies could be based on general equations like (2.5) for continuous-time systems or (2.9) for discrete-time systems, without the need to consider the particular properties of individual processes.

This general view leaded to the birth of Automatic Control as an independent discipline, the results of which are generic enough to be applied to many, apparently different, practical problems. Nevertheless, the ties of Automatic Control and its application domains remain. On the one hand, more and more Automatic Control engineers are in charge of developing practical control systems in different application areas. On the other hand, the practical applications and the new problems they reveal, indicate the requirements for further theoretical developments. It should be noted, however, that a notorious gap has always been claimed between theory and practice of Automatic Control, i.e. a certain inertia of penetration of theoretical results in various application domains.

The necessary steps when developing a control system for a particular application are typically the following:

1. **Modelling/Identification** is the phase of development of a mathematical process

model. The model may be deduced from according laws of physics, chemistry, etc. (deductive way); or it may be induced from experimental results (inductive way) showing the processes' response to selected input signals; or via a combination of both approaches. Often, it is necessary to derive more than one models for the same process, e.g. a simple control design model and a more accurate simulation model.

2. **Control System Configuration:** If not provided by the control problem, one has to select the variables, their locations, and the corresponding technologies for measurements and actuators. These decisions are neither easy to make nor negligible, as they may have a major impact on the control system performance, independently of the employed control strategy, particularly for large-scale processes.

3. **Control Strategy Design** is effectuated on the basis of the control design model, using the methodologies mentioned in the three previous sections.

4. **Simulation Test:** Particularly for complex processes, it is advisable, convenient, and cost-effective to test the control strategy via simulation experiments (or to compare the performance of various alternatives) for different scenarios before actual application.

5. **Implementation** of the control system including the measurement devices, the actuators, the communications and the control strategy. The latter is usually implemented in an analog or (more frequently) in a digital way (computer) within a decentralized, centralized, or hierarchical structure.

6. **Experimentation/Validation/Evaluation:** This phase aims at testing the proper functioning of the real control system (strategy, software, hardware, equipment), first in parts and then as a whole, and to evaluate its actual performance before entering the operational (and completely autonomous) phase.

2.7 The need for fine-tuning in control systems

In most practical cases, the six steps mentioned above must be partly iterated several times before the final system, with its desired performance characteristics, becomes operational. Even after this phase, elaboration of further (theoretically or practically based) improvements is quite common, particularly for complex and large-scale systems. In fact, the development of efficient control systems in areas like electric power distribution, communication networks, various military systems, various traffic control problems, requires the control system designer to acquire a profound understanding of the particular process behavior, the relevant

technologies, the operational objectives and constraints. In many cases, the forming of interdisciplinary development teams may be the most convenient approach.

For relatively simple systems, the mostly methodological steps 1, 3, 4, and partly 2 and 5 may be completed within a few days by experienced and knowledgeable Automatic Control engineers. On the contrary, for complex and/or large-scale systems, several person-years may be required for the successful completion of step 6 (experimentation/validation/evaluation). It is at this stage of the system design and deployment, that the **fine-tuning** procedure has to take place, in order to elevate the control algorithm to its best achievable performance.

Fine-tuning concerns the selection of appropriate (or even optimal) values for a number of design parameters included in the control strategy.

The need for this fine-tuning procedure is essential when a new control algorithm is implemented (or an operational but "aged" control algorithm needs to be updated) and has been found to be crucial for the overall system performance [37]. Typically, this fine-tuning procedure is conducted manually, via trial-and-error, relying on expertise and human judgment. Experienced engineers (in cooperation with practitioners or system operators) experiment with different sets of design parameters, trying to achieve an acceptable (close to optimal) system performance, according to some pre-specified performance metrics.

Finally, the results of the fine-tuning procedure of a well-designed control algorithm (e.g. urban signal control strategy), which is implemented in two different field applications (urban road networks), may sometimes lead to quite different sets of design parameters when compared to each other. A well-conducted system fine-tuning will result in "appropriate" values for the tunable parameters, minimizing a performance criterion, but these values may be "appropriate" only for the specific application. They may for example depend on the topology and/or the special characteristics of the studied urban road network. This is another reason that makes fine-tuning vital, even for a system that has been already fine-tuned but is going to be implemented in a new field application.

2.8 Examples of traffic control systems

As already mentioned, two fundamental characteristics of virtually any traffic system are the presence of a network structure and of some kind of flow therein. Based on both characteristics, it is not difficult to identify other domains of similar character:

- water or gas distribution networks,
- sewer networks,

- electric power distribution networks,
- communication networks (for telecommunication and/or data transfer),
- other transportation systems (air, maritime, rail traffic networks).

In all these processes with a meshed structure, a partially predictable and geographically distributed demand has to be satisfied by use of a corresponding large-scale network. In order to satisfy the demand, different kinds of flow control may be applied within the network, so as to minimize certain objective criteria subject to capacity and storage constraints. In communication networks, the capacity reduction during periods of strong demand is due to buffer overflows leading to the real or assumed loss of corresponding data packets. These packets are eventually re-sent from their respective origin nodes, leading to further increment of the demand and hence the buffer overflows, and so forth. In urban, air, rail, and partially maritime traffic networks, capacity reduction is mostly due to node blocking (junctions, airports) or due to a blocking of downstream links. Finally, in motorway networks, a self-blocking phenomenon as an inherent flow characteristic may become apparent, even within individual overloaded network links ("congestion from nothing"), independently of all other network links.

Although storage and queue-forming phenomena are common in most of these processes, the particular phenomenon of flow congestion that reduces the network's (or particular links') capacity and leads to a deteriorated infrastructure utilization, is mostly observed in communication and traffic flow networks of all kinds. In both cases, a common resource (the network infrastructure) is used competitively by many users. In this context, the appearance of a congestion that reduces the network's capacity at the time that it is most urgently needed (rush hour), is a paradox that has to be faced, reduced or avoided to the benefit of all users.

In urban and motorway traffic networks there are several examples of control procedures that can be applied:

- **Motorway networks:** ramp metering, route recommendation via variable message signs (VMS), driver information via VMS, lane control, surveillance, automatic incident detection (AID), other kinds of detection.
- **Urban networks:** signal control, individual route guidance, parking control.

In this thesis we concentrate on the real-time automated fine-tuning of two different existing traffic control systems. In Chapter 5 the developed adaptive fine-tuning (AFT) algorithm is applied to a large-scale ramp metering control system of a motorway, whereas in Chapter 6 the same algorithm is applied to a large-scale signal control fine-tuning problem of a complex urban road network.

Chapter 3

State of the art and related methodologies

This chapter presents the state of the art in parameter estimation/optimization methodologies. The content of the chapter is widely based on Spall's book [72], which provides a thorough review on this topic. Section 3.1 provides an introduction to the fundamental problem investigated throughout this thesis and Section 3.2 discusses some aspects about the general concept of stochastic search and optimization. Following, Section 3.3 provides an analytical presentation of Stochastic Approximation (SA) and analyzes the general principles of designing SA search algorithms. In Section 3.4 three popular SA algorithms are thoroughly described, namely FDSA, RDSA and SPSA. Finally, Section 3.5 concludes the chapter with some remarks about how to choose the appropriate algorithm for different applications.

3.1 Introduction

This chapter provides a review of algorithms and methodologies which attempt to solve two general – and closely related – mathematical problems. Let Θ be the domain of allowable values for a vector θ of dimension n_θ. The two main problems of interest are:

Problem 1: Find the values of a vector $\theta^* \in \Theta$ that minimize a scalar-valued loss function $J(\theta)$.

— or —

Problem 2: Find the values of $\theta^* \in \Theta$ that solve the equation $g(\theta^*) = 0$ for some vector-valued function $g(\theta)$.

In order for Problem 1 to have a solution we assume that domain Θ is compact and the loss function $J(\theta)$ is continuous in Θ. The vector θ represents a collection of tunable (or "adjustable") parameters that one is aiming to pick in the best way. The nonlinear loss function $J(\theta)$ is a scalar measure that summarizes the performance of the system for a given set of values of the tunable parameters. The domain Θ reflects allowable values (constraints) on the elements of θ and θ^* represents the optimal solution. Other common names for the loss function are performance function, objective function, fitness function, or criterion.

While Problem 1 above refers to minimizing a loss function, a maximization problem (e.g., maximizing profit) can be trivially converted to a minimization problem by changing the sign of the criterion. The nonlinear root-finding function $g(\theta)$ (which is generally a vector) often arises via calculating the gradient (derivative) of the loss function (i.e., $g(\theta) = \partial J(\theta)/\partial\theta$). More generally, $g(\theta)$ may represent a collection of functions that are derived from physical principles related to the system under study.

Versions of the two problems described above arise in countless areas of practice and engineering. Mathematical techniques of search and optimization are aimed at providing a formal means for making the best decisions in problems of the type above. Given the difficulties in many real-world problems and the inherent uncertainty in information that may be available for carrying out the task (the analytical form of functions $J(\theta)$, $g(\theta)$ is not always available), stochastic search and optimization methods have been playing a rapidly growing role. In many problems of practical interest, mathematical search algorithms – iterative procedures usually implemented on a computer – are used to produce a solution.

3.2 Stochastic search and optimization

The focus in this section is stochastic search and optimization. The meaning of "stochastic" is that the algorithms and methodologies considered here apply where (see Spall's book [72] for details):

Property A: There is random noise in the measurements of $J(\theta)$.

— and/or —

Property B: There is a random choice made in the search direction as the algorithm iterates toward a solution.

The above two properties contrast with classical deterministic search and optimization, where it is assumed that one has perfect information about the loss function (and derivatives,

if relevant), and that this information is used to determine the search direction in a deterministic manner at every step of the algorithm. In many practical problems, such information is not available, indicating that deterministic algorithms are inappropriate. Let $\theta(k)$ be the generic notation for vector θ at the k-th iteration of whatever algorithm is being considered, $k = 0, 1, 2, \ldots,$. Throughout this chapter, different mathematical forms for calculating $\theta(k)$ are presented, according to the algorithm being considered. In all these algorithms, it could be considered that $\theta(k)$ always represents a "random" vector, since it is derived from input under stochastic Properties A and/or B above.

For stochastic Property A, above, the noise is relative to the measurements of $J(\theta)$ (or sometimes even $g(\theta)$, if available). Relative to the second defining property of a stochastic algorithm, Property B above, it is sometimes beneficial to deliberately introduce randomness into the search process, as a means of speeding convergence and making the algorithm less sensitive to modeling errors. This "injected" randomness is usually created via computer-based pseudo-random number generators. Although the introduction of randomness may seem at first thought counterproductive, it is well known to have beneficial effects in some settings. One of the roles of "injected" randomness in stochastic search is to allow for "spontaneous" movements to unexplored areas of the search space, that may contain unexpectedly good values for vector θ. The randomness may provide the necessary "kick". This is especially relevant in seeking out a global optimum when the search is stalled near a local solution.

Closely related to this concept, is also the use of randomness in the important class of algorithms that emulate evolutionary principles of optimization; randomness is a central part of both physical and simulated evolution, through the introduction of mutations and through the choice of parents. These mutations may sometimes have a beneficial effect by allowing unexpected solutions to be evaluated. "Injected" randomness may also be used for the creation of simple random quantities that act like their deterministic counterparts, but which are much easier to obtain and more efficient to compute. An example of this, is the simultaneous perturbation approximation of a gradient vector in Section 3.4.4. This gradient approximation can be used in place of the true gradient in certain optimization schemes, yielding similar general performance. Also, the logic of AFT algorithm, presented in this thesis (see Chapter 4), is based on the same principle of "injected random perturbations".

Yet another area where "injected" randomness is useful is in numerical integration. Often, such methods are implementable when analytical methods are impractical or impossible. Such methods are usually more efficient in high-dimensional problems than deterministic quadrature approaches, provided that one is willing to tolerate a small probability of achieving a poor estimate. Markov chain Monte Carlo (see [24] for details), comprises a very popular

general approach for numerical integration via such "injected" randomness.

Some other principles of stochastic search and optimization, that have to be taken into account when developing an algorithm are the following:

- Relative efficiency via iterative function evaluations.
- Implications of noisy function measurements.
- "Curse of dimensionality".
- Near to optimal solutions versus optimal solution.
- Constraints.
- Stopping criteria.
- Time-varying problems.
- Uniqueness versus non-uniqueness of θ^*.

3.3 Stochastic approximation

3.3.1 Introduction

As Spall indicates in [72] a core approach in nonlinear stochastic search and optimization is stochastic approximation (SA), which corresponds to the problem of nonlinear estimation for solving nonlinear root-finding problems in the presence of noisy measurements. SA is a cornerstone of stochastic search and optimization as a generalization of the well-known deterministic algorithms (steepest descent and Newton-Raphson). The basic approach is sometimes referred to as the Robbins-Monro algorithm in honor of the two people who introduced the modern general setting.

SA was introduced in a 1951 article by Robbins and Monro [65], with important generalizations and extensions following close behind as given in Kiefer and Wolfowitz [32]. Originally conceived as a tool for statistical computation, it has come to thrive in numerous disciplines of electrical engineering. In control engineering, SA is the main paradigm for on-line algorithms for system identification and adaptive control. This is not accidental. The key word in most of these applications is **adaptive**. SA has several intrinsic characteristics that make it an attractive framework for adaptive schemes.

- It is designed for uncertain (stochastic) environments, where it allows one to track the "average" or "typical" behavior of such an environment.

- It is incremental, i.e., it makes small changes at each step, which ensures a graceful behavior of the algorithm. This is a highly desirable feature of any adaptive scheme.
- Furthermore, it usually has low computational and memory requirements per iterate, another desirable feature of adaptive systems.
- Finally, it conforms to our anthropomorphic notion of adaptation: it makes small adjustments so as to improve a certain performance criterion based on feedbacks received from the environment.

For these very reasons, there has been a resurgence of interest in this class of algorithms in several new areas of engineering. Some of them are communication networks (adaptive signal processing), artificial intelligence, neural networks, learning models in economics, and algorithms for reinforcement learning, a popular learning paradigm with applications in e-commerce, robotics, etc., (see, e.g., [5]). Finally, another major application domain of SA has provided the basis for many learning or "parameter tuning" algorithms in control engineering problems [7].

3.3.2 Stochastic steepest descent based methods

The method of steepest descent is one of the oldest formal optimization techniques. Nevertheless, it remains one of the more popular deterministic approaches. For example, it corresponds to the widely used backpropagation algorithm for neural networks when one is working with a fixed set of input-output data. Steepest descent is based on the simple principle that from a given value θ the best direction to go is the one that produces the largest local change in the loss function $J(\theta)$ (the steepest descent). The gradient vector $g(\theta)$ at the given θ defines this direction. Hence the algorithm is

$$\theta(k+1) = \theta(k) - \alpha(k)g(\theta(k)) \tag{3.1}$$

where k is the iteration count, $\theta(0)$ is the initial "guess" about likely values of θ^*, and $\alpha(k)$ is the stepsize, which may be specified a priori (often as a constant $\alpha(k) = \alpha$) or picked on an iteration-to-iteration basis as a solution to

$$\min_{\alpha(k)\geq 0} \{J(\theta(k)) - \alpha(k)g(\theta(k))\}. \tag{3.2}$$

This secondary optimization problem is called a line search. So, (3.1) states that the new estimate of the best value of θ is equal to the previous value minus a term proportional to the gradient at the current value.

All the stochastic methods that are presented in the following sections are based on the general principle of the gradient descent method. Based on this simple principle (follow the

direction of the gradient), different researchers have developed many stochastic analogues of steepest descent. All these methods, typically have a predetermined decaying sequence $\alpha(k)$ which plays a critical role in the algorithm, often determining whether the algorithm converges or diverges. Aside from being called stepsizes, these factors are sometimes referred to as gains or learning coefficients, depending on the field of application. Conditions guaranteeing that the steepest descent iterate converges to θ^* as $k \to \infty$ are presented in many places (e.g., [4]).

3.3.3 General principles of stochastic approximation methods

A central aspect of SA is the allowance for noisy input information in the algorithm. In fact, SA methods presented in this chapter are often better at coping with noisy input information than other existing search methods. Moreover, the theoretical foundation for SA is deeper than the theory for other stochastic search methods with noisy measurements. In the case of root-finding SA, the noise manifests itself in the measurements of $g(\theta)$ used in the search procedure as θ varies. More specifically, suppose that noisy measurements of $g(\theta)$ at any $\theta(k)$ are available as

$$\widehat{g}(\theta(k)) = g(\theta(k)) + e(k), \quad k = 0, 1, 2, \ldots, \tag{3.3}$$

where $g(\theta(k))$ is the real gradient of function $J(\theta)$ at point $\theta(k)$, $\widehat{g}(\theta(k))$ the noisy measurement of $g(\theta(k))$ and $e(k)$ is assumed to be some noise term of dimension n_θ at the k-th iteration of the algorithm. So, for a specified θ, a noisy measurement $\widehat{g}(\theta)$ of the gradient is returned. One can show, that there exist $g(\theta)$ and $e(k)$ such that measurements of form (3.3) yield a solution to Problem 2 defined in Section 3.1. For example, if $E[e(k)] = 0$ (something that is more general than it appears), then $g(\theta(k)) = E[\widehat{g}(\theta(k))]$.

Based on the basic steepest descent algorithm, an obvious implementation with noisy measurements of $g(\theta)$ is to average many measurements of the form (3.3) in every iteration k. Such averaging is used to approximate $g(\theta(k))$ from multiple values of $\widehat{g}(\theta(k))$. A significant innovation of Robbins and Monro [65] was the recognition that this is a "wasteful" use of the measurements. Recall that $g(\theta(k))$ is merely an intermediate calculation towards the ultimate goal of trying to find a root θ^*. There is little interest in $g(\theta(k))$ itself. So, the main innovation in SA is to do a form of averaging across iterations. At first thought, this type of averaging may seem dubious, since the underlying evaluation point θ is changing across iterations. Albeit, as suggested by Robbins and Monro [65], this across-iteration averaging can lead to a more effective use of the input information, than expending a large amount of resources in getting accurate estimates for $g(\theta)$ at each iteration.

Historically, stochastic approximation started as a scheme for solving the Problem 2

defined in Section 3.1, given "noisy measurements" $\widehat{g}(\theta)$ of the function $g(\theta)$. That is, we are given a black box which on input θ gives as its output $\widehat{g}(\theta) = g(\theta) + e$, where e is a random variable representing noise. The stochastic approximation scheme proposed by Robbins and Monro [65] was to run the iteration

$$\theta(k+1) = \theta(k) - \alpha(k)\left[g(\theta(k)) + e(k)\right] \tag{3.4}$$

where $e(k)$ is the noise sequence and $\alpha(k)$ are positive scalars satisfying certain constraints (see Subsection 4.7.1). The expression in the square brackets on the right is the noisy measurement. That is, $g(\theta(k))$ and $e(k)$ are not separately available, only their sum $\widehat{g}(\theta(k))$ is. The above root-finding algorithm is clearly motivated by the steepest descent algorithm. Since its inception, the scheme (3.4) has been a cornerstone in scientific computation. This has been so largely because of the following advantages:

- It is designed to handle noisy situations. One may say that it captures the average behavior in the long run. The noise in practice may not only be from measurement errors or approximations, but may also be added deliberately as a probing device or a randomized action, as, e.g., in certain dynamic game situations.
- It is incremental, i.e., it makes small moves at each step. This typically leads to more graceful behavior of the algorithm at the expense of its speed.
- In typical applications, the computation per iterate is low, making its implementation easy.

These three features make the scheme (3.4) ideal for applications where the key word is "adaptive".

Another major innovation of SA algorithms (that use a scheme similar to (3.4)) is the specification of precise conditions on the gain coefficients $\alpha(k)$, in order to ensure that the process properly invokes the across-iteration averaging and converges to a root θ^*. As expected, these conditions generally differ from those in the easier deterministic steepest descent setting (see Subsection 4.7.1). Of course, these conditions also apply in the deterministic steepest descent algorithm because that is a special case of SA.

3.4 Existing gradient-free methodologies

The previous section introduced the root-finding (Robbins-Monro) SA algorithm as a general method for nonlinear problems. The aim is to find one or more zeros of the function $g(\theta)$ (i.e., roots of $g(\theta) = 0$) when only noisy measurements of the function $g(\theta)$ are available. There are,

however, a large number of problems where the direct measurement of $g(\theta)$ (which depicts the gradient $\partial J/\partial\theta$), is difficult or impossible to obtain. For this reason, there is considerable interest in SA algorithms that do not depend on direct gradient measurements. Rather, these algorithms are based on an **approximation** of the gradient formed from (generally noisy) measurements of function J.

This interest has been motivated, for example, by problems in adaptive control, statistical identification of complex systems, optimization of processes by Monte Carlo simulations, training of recurrent neural networks, design of complex queuing and discrete-event systems, and applications of model-free feedback control systems. In contrast to the stochastic gradient algorithm of Robins and Monro, the methods discussed in this section are gradient-free. The use of "gradient-free" is associated with the fact that the implementation of the algorithms does not require any knowledge about the actual gradient $g(\theta)$.

Now, let return to the essential optimization problem of minimizing a loss function $J(\theta)$, as defined in Problem 1 of Section 3.1. It is assumed that the stochastic gradient $(\partial J/\partial\theta)$ is not available and only (generally noisy) measurements of the loss function are available. In this case, we may express the loss function $J(\theta)$ as

$$J(\theta) = E\left[\widehat{J}(\theta, e)\right] \tag{3.5}$$

where e represents the random effects in the process generating the system output and $\widehat{J}(\theta, e)$ represents some "observed" cost as a function of the chosen θ and random effects e. The variable e may represent the amalgamation of many individual random effects. The expectation in equation (3.5) is with respect to all randomness embodied in e. So $J(\theta)$ represents an average cost over all possible values of e at the specified θ.

Because of the nonlinearity, or possible lack of knowledge of the analytical form of the loss function (and the probability distribution of e), it is almost never the case that $J(\theta)$ can be computed. However, $\widehat{J}(\theta, e)$ is typically available since it just represents the outcome for a particular experiment (no expectation involved). Furthermore, for all mathematical proves of the considered methodologies, it is assumed that $\widehat{J}(\theta, e)$ is a differentiable function (in θ) for almost all e (i.e., for all values of e except possibly a set of values having probability zero). The expression $\partial\widehat{J}(\theta, e)/\partial\theta$ is called a stochastic gradient because it depends on the random term e.

3.4.1 The Kiefer-Wolfowitz algorithm (FDSA)

The oldest method for gradient approximation is the finite-difference (FD) approximation, which relies on small one-at-a-time changes applied to each of the individual elements of θ. After each change, the (possibly noisy) value of $J(\theta)$ is measured. When measurements

of $J(\theta)$ have been collected for perturbations in each of the elements of θ, the gradient approximation may be formed. The FD approach is motivated directly from the definition of the gradient as a collection of derivatives for each of the components in θ, holding all other components fixed. In fact, the algorithm has the even weaker requirement of only requiring measurements of the difference of two values of the loss function, as opposed to measuring the loss function itself.

Unfortunately, the method can be very costly if the dimension n_θ is high, since one must collect at least one $J(\theta)$ measurement for each of the elements in θ. This cost motivates the Monte Carlo-based approaches discussed in the next two sections. Nevertheless, the FD method is fundamental in both stochastic and deterministic optimization.

Building on the seminal Robbins and Monro paper [65], an SA algorithm based on the FD gradient approximation was introduced for scalar θ in Kiefer and Wolfowitz [32] and multivariate θ in [6]. The FD-based SA (FDSA) algorithm is the oldest SA method using gradient approximations built only from loss measurements. Because of its relative ease of use, FDSA is much more widely used in practice than the stochastic gradient-based methods in at least one important area – simulation-based optimization ([20]). The recursive procedure used here is in the general SA form

$$\theta(k+1) = \theta(k) - \alpha(k)\widehat{g}(\theta(k)) \tag{3.6}$$

where $\widehat{g}(\theta(k))$ is the estimate of the true gradient $g(\theta(k))$ of the loss function $J(\theta)$ at the iteration k, based on measurements of function J. Hence, (3.6) is analogous to the stochastic gradient algorithm, with the gradient estimate $\widehat{g}(\theta(k))$ replacing the direct gradient measurement $\widehat{g}(\theta) \equiv \partial\widehat{J}/\partial\theta$ at $\theta = \theta(k)$. The gain $\alpha(k) \geq 0$ here acts in a way similar to its role in the stochastic gradient form. Under appropriate conditions, the iteration in (3.6) converges to θ^* in some stochastic sense (usually almost surely, a.s.).

The essential part of (3.6) is the gradient approximation $\widehat{g}(\theta(k))$. We discuss below the oldest and best-known means of forming the approximation – the FD method. Expression (3.6) with this approximation represents the FDSA algorithm. One-sided gradient approximations involve measurements $\widehat{J}(\theta(k))$ and $\widehat{J}(\theta(k)+\text{perturbation})$, while two-sided approximations involve measurements of the form $\widehat{J}(\theta(k)\pm\text{perturbation})$. The two-sided FD approximation for use with (3.6) is

$$\widehat{g}(\theta(k)) = \begin{bmatrix} \frac{\widehat{J}(\theta(k)+c(k)\Delta_1)-\widehat{J}(\theta(k)-c(k)\Delta_1)}{2c(k)} \\ \frac{\widehat{J}(\theta(k)+c(k)\Delta_2)-\widehat{J}(\theta(k)-c(k)\Delta_2)}{2c(k)} \\ \vdots \\ \frac{\widehat{J}\left(\theta(k)+c(k)\Delta_{n_\theta}\right)-\widehat{J}\left(\theta(k)-c(k)\Delta_{n_\theta}\right)}{2c(k)} \end{bmatrix} \tag{3.7}$$

where Δ_i, denotes a vector with a 1 in the i-th place and O's elsewhere and $c(k) > 0$ defines the difference magnitude. The pair $\{\alpha(k), c(k)\}$ are the gains (or gain sequences) for the FDSA algorithm. An obvious analogue to (3.7) holds for the one-sided FD approximation. In this case, the i-th component of the gradient approximation is

$$\frac{\widehat{J}\left(\theta(k) + c(k)\Delta_i\right) - \widehat{J}(\theta(k))}{c(k)}. \tag{3.8}$$

The two-sided form in (3.7), called FDSA algorithm, is the obvious multivariate extension of the scalar two-sided form in Kiefer and Wolfowitz [32]. The initial multivariate method in [6] used a one-sided approximation.

3.4.2 Extensions of the FDSA algorithm

References [16], [17] present several methods for accelerating the convergence of FDSA-type algorithms, analogous to some of the methods for root-finding SA. These methods are based on taking additional measurements to explore the loss function surface in detail. One such method in [17] is a stochastic analogue to second-order algorithms of the generic Newton-Raphson form. This algorithm uses $O(n_\theta^2)$ measurements of $\widehat{J}(\cdot)$ per iteration for the gradient and Hessian estimation. The gradient is estimated in a standard way (e.g., (3.7)) and the Hessian is estimated using an analogous FD-based double-differencing scheme. Although this method is intuitively sensible, it demands many extra loss measurements if n_θ is even moderately large and is likely to be numerically unstable with even a small level of noise.

A more systematic approach has been adapted under the rubric of perturbation analysis (PA). In this approach, one looks for ways to get a gradient estimate, at any θ value, based on only one or a small number of simulation runs. Given the stochastic nature of the simulation, this gradient estimate is only a stochastic estimate of the true gradient $g(\theta) = J/\partial\theta$. A specific form of PA is the infinitesimal perturbation analysis (IPA) approach to generating a gradient estimate. IPA requires that the probability distribution generating e be independent of θ, consistent with the standard formulation for the stochastic gradient method discussed in Section 3.3.3. Thus, the IPA gradient estimate has the generic form $\partial\widehat{J}\left(\theta, e\right)/\partial\theta$, as appropriate. The IPA method has a significant potential advantage in efficiency over traditional methods based on approximating the gradient using finite-difference methods. The finite-difference methods typically require between $n_\theta + 1$ and $2n_\theta$ simulation runs to form a gradient approximation, in contrast to the one run for IPA (or a small number of runs).

3.4.3 The Random Directions Kiefer-Wolfowitz (RDSA)

"Injected" Monte Carlo randomness – as in Property B in Section 3.2 – can be used in combination with FDSA-type methods. This is also the essence of the developed algorithm presented in Chapter 4. Ermoliev [15] was apparently the first to introduce such an idea via random directions SA (RDSA). The essential idea with RDSA is to use the basic SA recursion in (3.6), but to replace the FD gradient approximation in (3.7) with a more efficient gradient approximation, which is generated with the help of a Monte Carlo-generated perturbation vector. The basic form of this approach requires only two loss measurements to approximate the gradient vector (for any dimension n_θ) and replaces the deterministic perturbations of FDSA (i.e., the $\pm c(k)\Delta_i$ at each k for all $i = 1, 2, \ldots, n_\theta$) with **random perturbations**.

Relative to classical two-sided FDSA (equations (3.6) and (3.7)), RDSA provides the potential for increased efficiency because of the n_θ-fold reduction in loss measurements per iteration ($2n_\theta$ for FDSA versus 2 for RDSA). Koronacki, [35], also considers the idea of random perturbations, including some convergence theory. This paper suggests the use of $2m$ loss measurements, with m usually less than n_θ. There is, however, no formal evidence of improved efficiency over basic FDSA.

The basic form for the i-th component of the RDSA gradient approximation at iteration k is

$$\widehat{g}_i(\theta(k)) = \pi_i(k)\frac{\widehat{J}\left(\theta(k) + c(k)\pi(k)\right) - \widehat{J}\left(\theta(k) - c(k)\pi(k)\right)}{2c(k)} \tag{3.9}$$

where $\pi(k) = [\pi_1(k), \pi_2(k), \ldots, \pi_{n_\theta}(k)]^T$ is a vector of Monte Carlo-generated random variables satisfying certain regularity conditions (see [72]). Note that the two $J(\cdot)$ values are reused for all elements of the gradient approximation. The choice of probability distribution for the Monte Carlo perturbations is important in realizing the desired improvements in efficiency. Ermoliev, in [15] includes analysis of the bias in the gradient approximation and considers one specific distribution (uniform) for the $\pi_i(k)$.

In conclusion, one of the main shortcomings of FDSA is its inefficiency in high-dimensional problems. That is, the number of loss measurements in each gradient approximation grows directly with the dimension. As a typical implementation for optimization requires many gradient approximations (one at each iteration), the overall number of loss measurements in the optimization process may become prohibitive. To tackle this problem, some methods have been introduced that create gradient approximations via Monte Carlo schemes. By reducing the number of loss measurements for each gradient approximation, these methods may offer significant improvements in efficiency.

3.4.4 Simultaneous Perturbation Stochastic Approximation

Continuing in the spirit where only noisy loss measurements are available, this section discusses the simultaneous perturbation stochastic approximation (SPSA) algorithm for stochastic optimization of multivariate systems [69]. Relative to the finite-difference-based methods, the principal benefit of SPSA is a reduction in the number of loss measurements required to achieve a given level of accuracy in the optimization process. The central focus with SPSA is the stochastic setting where only measurements of the loss function are available (i.e., no gradient information).

The basic unconstrained SPSA algorithm is in the general recursive SA form:

$$\theta(k+1) = \theta(k) - \alpha(k)\widehat{g}(\theta(k)) \tag{3.10}$$

where $\widehat{g}(\theta(k))$ is the simultaneous perturbation estimate of the gradient $g(\theta) = \partial J/\partial\theta$ at the k-th iteration (based on the measurements of the loss function) and $\alpha(k)$ is a nonnegative scalar gain coefficient.

The essential part of (3.10) is the gradient approximation $\widehat{g}(\theta(k))$. Recall that with FDSA, this gradient approximation is formed by perturbing the components of $\theta(k)$ one-at-a-time and collecting a loss measurement $\widehat{J}(\cdot)$ at each of the perturbations (in practical problems, the loss measurements are sometimes noise-free, i.e., $\widehat{J}(\cdot) = J(\cdot)$). This requires $2n_\theta$ loss measurements for a two-sided FD approximation. In contrast, with simultaneous perturbation, all elements in $\theta(k)$ are randomly perturbed together to obtain two loss measurements $\widehat{J}(\cdot)$. For the two-sided SP gradient approximation, this leads to

$$\begin{aligned}\widehat{g}(\theta(k)) &= \begin{bmatrix} \frac{\widehat{J}(\theta(k)+c(k)\Delta(k))-\widehat{J}(\theta(k)-c(k)\Delta(k))}{2c(k)\Delta_1(k)} \\ \frac{\widehat{J}(\theta(k)+c(k)\Delta(k))-\widehat{J}(\theta(k)-c(k)\Delta(k))}{2c(k)\Delta_2(k)} \\ \vdots \\ \frac{\widehat{J}(\theta(k)+c(k)\Delta(k))-\widehat{J}(\theta(k)-c(k)\Delta(k))}{2c(k)\Delta_{n_\theta}(k)} \end{bmatrix} \\ &= \frac{\widehat{J}\left(\theta(k)+c(k)\Delta(k)\right) - \widehat{J}\left(\theta(k)-c(k)\Delta(k)\right)}{2c(k)}[\Delta(k)]^{-1},\end{aligned} \tag{3.11}$$

where $\Delta(k) = [\Delta_1(k), \Delta_2(k), \ldots, \Delta_{n_\theta}(k)]^T$ is a zero-mean n_θ-dimensional random perturbation vector, which has a user-specified distribution satisfying certain conditions (see [72] for details), $[\Delta(k)]^{-1} = \left[\Delta_1^{-1}(k), \Delta_2^{-1}(k), \ldots, \Delta_{n_\theta}^{-1}(k)\right]^T$, and $c(k)$ is a positive scalar. Because the numerator is the same in all n_θ components of $\widehat{g}(\theta(k))$, the number of loss measurements needed to estimate the gradient in SPSA is two, regardless of the dimension n_θ. Recall that RDSA algorithm uses a similar random directions gradient approximation.

While the number of loss function measurements $\widehat{J}(\cdot)$ needed in each iteration of FDSA grows with n_θ, the number in SPSA is fixed. This measurement savings per iteration, of

course, provides only the **potential** for SPSA to achieve large savings (over FDSA) in the total number of measurements required to estimate θ^* when n_θ is large. This potential is realized if the number of iterations required for effective convergence to an optimum θ^* does not increase in a way to cancel the measurement savings per gradient approximation at each iteration. We would expect this potential to be realized if, roughly speaking, the FD and SP gradient approximations acted the same in some statistical sense relative to their use in the basic optimization recursion (which is the same basic form in both FDSA and SPSA).

It is clear that the SP approximation above will not act the same as the FD approximation as an estimate of the gradient. The FD approximation will generally be superior in that sense. However, the interest is not in the gradient itself. Rather, the interest is in how the approximations operate when considered in optimization over multiple iterations with a changing point of evaluation θ. Spall (in [72]) studies the efficiency of both algorithms in various problems, establishing the fundamental result: under reasonably general conditions (see [72] for details), the SPSA and FDSA recursions achieve the same level of statistical accuracy, for a given number of iterations, even though SPSA uses only $1/n_\theta$ times the number of function evaluations of FDSA (since each gradient approximation uses only $1/n_\theta$ the number of function evaluations).

The informal rationale for the strange-looking gradient approximation in (3.11) is quite simple. It can be shown (see [72] for the mathematical proof), using a simple Taylor expansion, that this approximation is an "almost unbiased" estimator of the true gradient (considering that the measurement noise $e(k)$ has mean zero and that J is several times differentiable).

A one-measurement form of the SP gradient approximation is considered in [71]. The gradient approximation has the form

$$\widehat{g}(\theta(k)) = \begin{bmatrix} \frac{\widehat{J}(\theta(k)+c(k)\Delta(k))}{c(k)\Delta_1(k)} \\ \frac{\widehat{J}(\theta(k)+c(k)\Delta(k))}{c(k)\Delta_2(k)} \\ \vdots \\ \frac{\widehat{J}(\theta(k)+c(k)\Delta(k))}{c(k)\Delta_{n_\theta}(k)} \end{bmatrix}. \tag{3.12}$$

Although the form above may seem strange in that it does not include explicit information related to the difference of function values, the form shares the nearly unbiased property of the standard two-measurement form in (3.11). In particular, via a Taylor expansion of $\widehat{J}\left(\theta(k)+c(k)\Delta(k)\right)$, it is found that $E\left[\widehat{g}(\theta(k))\right] = g\left(\theta(k)\right) + O(c^2(k))$. Although it is shown in [71] that the standard two-measurement form is usually more efficient (in terms of the total number of loss function measurements to obtain a given level of accuracy in the θ iterate), there may be advantages to the one-measurement form in real-time operations. Such real-

time applications include target tracking and feedback control, where the underlying system dynamics may change too rapidly to get a credible gradient estimate with two successive measurements. In the simulation experiments of Chapter 5 and Chapter 6 a variant version of the one-measurement SPSA is used, proposed for real-time applications (see equations (5.9), (5.10) in Subsection 5.6.1 for the formula definition).

The problem of constrained (equality and inequality) optimization with SPSA is considered in [66] and [19] using a projection approach. The projection algorithm reads

$$\theta(k+1) = H_{\Theta}\left[\theta(k) - \alpha(k)\widehat{g}(\theta(k))\right], \tag{3.13}$$

where $H_{\Theta}[\cdot]$ is the mapping that projects any point not in the constraint domain Θ to a new point inside Θ. While the projection approach has an elegant mathematical form, it is quite restricted in the types of constraints that can be handled in practical problems. Essentially, the constraints must be represented explicitly in a "nice" way, so as to facilitate the mapping of a constraint violation in θ to the nearest valid point. A common implementation of projections is to problems with hypercube constraints, where the individual components of θ are bounded above and below by user-specified constants.

3.5 Concluding remarks

The stochastic approximation framework introduced in Section 3.3 is a powerful approach for dealing with nonlinear root-finding and optimization problems, especially problems involving noisy measurements of the involved functions. The chapter has continued with the SA-based optimization setting emphasized in Section 3.4, but with the significant difference that direct measurements of the gradient are not required. Rather, this chapter has considered methods that build gradient approximations from measurements of the loss function. This **mathematical** distinction has profound **practical** implications (again, Spall's book [72] provides a detailed analysis).

In particular, it is often difficult or impossible to calculate the stochastic gradient for the root-finding methods. We discussed three gradient-free SA search algorithms (FDSA, RDSA, SPSA) which are fundamentally based on loss function values (no gradients). Although they require only loss function measurements, gradient-free SA algorithms exhibit convergence properties similar to the properties of the stochastic gradient methods. All three algorithms have rigorous mathematical proves for convergence (under certain assumptions). The indirect connection to the gradient usually enhances the convergence when there is not a great danger of converging to an unacceptable local minimum. Finally, it should be noted, that although the presented SA search methods have a deep theoretical justification with noisy

measurements, they are fundamentally local optimizers.

Direct measurements of the gradient $J/\partial\theta$ do not typically arise naturally in the course of operating or simulating a system. Hence, one must have detailed knowledge of the underlying system input-output relationships in order to calculate the $J/\partial\theta$. In contrast, the SA approaches require only conversion of the basic output measurements to sample values of the loss function, which does not require full knowledge of the system input-output relationships. Because of the fundamentally different information needed in implementing the stochastic gradient-based and gradient-free algorithms, it is difficult to construct meaningful methods of comparison. As a general rule, however, the stochastic gradient algorithms converge faster when speed is measured in number of iterations. This is not surprising given the additional information required for the gradient-based algorithms.

In practical applications, many factors besides the asymptotic rate of convergence must be considered in determining which algorithm is most appropriate for a given circumstance. The stochastic gradient algorithms may be either infeasible (if no system model is available) or undependable (if a poor system model is used). Furthermore, the total cost to achieve effective convergence depends not only on the number of iterations required, but also on the cost per iteration, which is typically greater in gradient-based algorithms. This cost may include greater computational burden, additional human effort required for determining and writing software for gradients, and experimental costs for model building. Finally, the rates of convergence are based on asymptotic theory, and may not represent practical convergence rates in finite-samples. As a general rule, however, if direct (stochastic) gradient information is conveniently and reliably available, it is generally to one's advantage to use this information in the optimization process. This thesis, however, focuses on the setting where such information is not readily available.

Another issue of major importance in gradient-free methods is the selection of the gain sequences $\alpha(k)$ and $c(k)$. In practise, the gains are usually chosen by trial and error on some small-scale (e.g., reduced number of iterations) version of the full problem. Generally, slowly decaying gains are preferred in case of noisy measurements. For picking a good stepsize $\alpha(k)$ one should balance the possibility of avoiding instabilities in the early iterations, in contrast to the possibility of non convergence in the later iterations (problem that is not easy to solve in many practical situations).

To cope with noise effects, in some problems is effective to set $c(k)$ at a level approximately equal to the standard deviation of the measurement noise in $J(\theta)$. This helps keep the n_θ elements of the gradient estimation $\widehat{g}(\theta)$ from getting excessively large in magnitude, before $\alpha(k)$ has decreased enough to compensate during the search process (the standard deviation of the noise can be estimated by collecting several $J(\theta)$ values for the same θ). However, if

the standard deviation changes dramatically with θ, this approach might not be useful. In the case where one has perfect (noise-free) measurements of $J(\theta)$, then $c(k)$ should be chosen as some small positive number.

Chapter 4

The Adaptive Fine-Tuning (AFT) algorithm

This chapter explores the AFT algorithm. Section 4.1 is a brief introduction to the relation of SA algorithms to the fine-tuning of control systems. Section 4.2 provides a definition of the problem in hand. Section 4.3 discusses the use of theoretical/simulation-based methods as a solution to this problem and Section 4.4 describes the general perspectives of adaptive and neural/learning techniques. Section 4.5 focuses on universal approximators and presents two popular methods about choosing the appropriate approximator for an application. In Section 4.6 the step-by-step application of AFT algorithm is thoroughly presented. Section 4.7 discusses efficient techniques about calculating stepsizes for SA methods. Finally, Section 4.8 presents some requirements that the studied system should meet, in order for the AFT algorithm to converge.

4.1 Introduction

The previous chapter discussed the use of stochastic approximation (SA) for problems of minimizing a loss function $J(\theta)$. Both cases where direct unbiased measurements of the gradient $g(\theta)$ are available (root-finding case), and where the optimization is carried out only with noisy measurements of the loss function $J(\theta)$ (gradient-free case) were considered. In this chapter we formulate the methodologies discussed in Chapter 3 to the problem of fine-tuning of large-scale nonlinear systems. Fine-tuning is a problem similar to the one defined in Section 3.1, as one is aiming to pick parameters θ (the system's design tunable parameters) in the best way, in order to optimize some scalar performance function $J(\theta)$ (the overall control system performance).

As a matter of fact, parameter estimation and fine-tuning are two closely related problems.

The same family of methods (SA search and optimization algorithms) can be applied to both problems, as they seek for optimal solution under system uncertainties (noise measurements). The iterative adaptive scheme of these algorithms plays also a very important role during the process of trying to adjust the design parameters of a control system. The system's performance has to be evaluated under different sets of design parameters and one should try to optimize this performance based on the noisy (measured) system outputs.

Finally, it should be noted that FDSA and RDSA methodologies are not readily appropriate to apply to online control system fine-tuning. For large-scale problems (where the dimension n_θ is high) an inhibited number of simulation runs is needed, in order for these methodologies to form an estimate of the gradient $g(\theta)$. On the other hand, SPSA method requires only one or two measurements per iteration. This is an important property that makes SPSA appropriate for application in such control calibration approaches.

4.2 Problem formulation

This section presents the general mathematical formulation for the problem of control systems fine-tuning. Consider a general discrete-time control system where the underlying dynamics are described according to the following nonlinear first-order difference equation

$$z(t+1) = F\left(z(t), u_i(t), d(t), t\right), \quad z(0) = z_0 \tag{4.1}$$

where $z(t), u_i(t), d(t)$ are the vectors of system states, control inputs, and exogenous (possibly measurable) signals, respectively, t denotes the discrete time-index, i denotes the regulator-index and $F(\cdot)$ is a sufficiently smooth nonlinear vector function. Note that the proposed methodology can be applied to a system even if the function F is **unknown**.

Consider also, that one or more control laws are applied to the system (4.1), which are described as follows:

$$u_i(t) = \varpi_i\left(\theta_i, z(t)\right) \tag{4.2}$$

where $\varpi_i(\cdot)$ are known smooth vector functions and θ_i is the vector of the tunable parameters for the i-th regulator. Note that we do not impose any restriction neither on the form of the equation (4.2), nor on the number of the applied control laws. Also, the discrete time-index t may be different for each control law i.

The overall system performance is evaluated through the following objective function

(performance index)

$$J\left(\theta; z(0), D_T\right) = \pi_T(z(T)) + \sum_{i=1}^{I}\sum_{t=0}^{T-1} \pi_{i,t}\left(z(t), u_i(t)\right)$$
$$= \pi_T(z(T)) + \sum_{i=1}^{I}\sum_{t=0}^{T-1} \pi_{i,t}\left(z(t), \varpi\left(\theta_i, z(t)\right)\right) \tag{4.3}$$

where $\theta = \text{vec}\left(\theta_1, \theta_2, \ldots, \theta_I\right)$, π_T and $\pi_{i,t}$ are known non-negative functions, I is the number of the fine-tuned regulators, T the finite time-horizon over which the control laws (4.2) are applied and $D_T \stackrel{\triangle}{=} [d(0), d(1), \ldots, d(T-1)]$ denotes the time-history of the exogenous signals over the optimization horizon T. By defining $x = \text{vec}\left(z(0), D_T\right)$, equation (4.3) may be rewritten as

$$J\left(\theta; z(0), D_T\right) = J\left(\theta, x\right). \tag{4.4}$$

Equation (4.4) indicates that the system performance is affected by the vector of the system's tunable parameters θ and the exogenous vector x. The problem in hand is to develop an appropriate iterative algorithm, which will be applied every T and will update the current control system parameters vector θ, so as to achieve better performance but also provide safe and efficient behavior. This means, that the algorithm should guarantee the stable and sustainable system performance.

In every iteration k of the algorithm (fine-tuning experiment) the following are taking place:

- the LNTCS performance (4.1)–(4.3) is evaluated for $\theta = \theta(k)$ through the measurement

$$J(k) \equiv J(\theta(k), x(k)) \tag{4.5}$$

- the current vector with the regulator's design parameters $\theta(k)$ is updated, so that it converges – as close as possible – to one of the local minima θ^* of the average value of J (with respect to the exogenous random vectors $x(k)$), defined according to

$$E\left[\frac{\partial J}{\partial \theta}\left(\theta^*, x(k)\right) | \mathcal{G}(k)\right] = 0 \tag{4.6}$$

  where $\mathcal{G}(k)$ is an appropriately defined term referring to the past values of vector θ and exogenous inputs x.

The requirement for convergence of $\theta(k)$ to one of the local minima θ^* is not sufficient in most practical situations; additionally to this requirement, the fine-tuning algorithm should be able to provide with **safe and efficient** performance during the fine-tuning process. More

precisely, at each iteration of the fine-tuning algorithm k, the performance index measurement should satisfy

$$J(k) \leq J(k-1) + \epsilon(k) \tag{4.7}$$

where $\epsilon(k)$ is an appropriately defined positive term, whose magnitude is proportional to the magnitude and variance of the exogenous inputs.

The requirement (4.7) is more than crucial in most practical LNTCS fine-tuning applications, since violation of such requirement may cause serious performance, safety, etc., problems. For instance, in the case of traffic control systems fine-tuning, the violation of requirement (4.7) may lead to serious problems (e.g., complaints, dangerous driving, etc.) that may force the traffic operators to cancel the fine-tuning process; similarly, in the case of LNTCS fine-tuning for mechanical structures, the violation of this requirement may cause the permanent deformation or even the destruction of the structure. It is worth noting, that standard AO methodologies such as the SPSA algorithm cannot guarantee that the requirement (4.7) holds during the fine-tuning process, mainly due to the use of random perturbations applied to the regulator parameters.

4.3 Theoretical/simulation-based methods

The last decades, attempts have been made in particular LNTCS applications to develop model-based, i.e., either theoretical-based or simulation-based designs that produce "good" sets of tunable parameters. Although they have helped in some cases to reduce time and effort for installation and maintenance, they did not manage to eliminate, or at least reduce significantly, the involvement of the human factor. One example in this class is the implementation of a Variable Speed Limit (VSL) system on the UK motorway M42 [52]. Despite the fact that the initial tunable parameters of the system (which correspond to speed and flow activation/deactivation thresholds) were "optimized" using theoretical tools derived from traffic flow theory and extensive simulation experiments, it took more than a year of calibration of the aforementioned thresholds until the system reached an acceptable performance. During this initial deployment phase the system performance was sometimes worse than in the no-control case.

There have also been some attempts to incorporate optimization-based tools within the maintenance procedure, see e.g. [45], [67], [31], [46] for an indicative list of references. In these cases, the problem of providing efficient maintenance is formulated as an optimization problem, where the tunable LNTCS parameters are chosen so as to optimize a performance criterion (e.g. average network speeds in traffic networks, average delays in airborne or seaborne transport systems, total number of containers loaded/unloaded in seaport con-

tainer terminals, average deviation from the operational schedules in public transport systems). However, optimization of such a performance criterion requires perfect or, at least, very accurate knowledge of the transport system dynamics as well as the demand.

To deal with this problem, optimization-based approaches employ simulation-based or theoretical models for representing the actual system dynamics. Then, based on the assumption that these models represent quite accurately the actual LNTCS operations, different optimization algorithms (e.g. gradient-descent, Gauss-Newton, evolutionary programming or neural network-based optimization algorithms) are applied in order to extract the optimal values of the tunable parameters. However, these approaches (a) require extensive and continuous calibration of the simulation/theoretical-based models, so as to optimize their approximation accuracy with respect to the actual transport system operations, and (b) face the tradeoff between simplicity and accuracy; in most cases, accuracy has to be sacrificed in order to avoid the use of extreme computational requirements of simulation or mathematical models that employ detailed modeling of the LNTCS operations.

4.4 Adaptive and neural/learning methods

One possible way to by-pass the above-mentioned problems is to incorporate adaptive or adaptive-like designs (such as neural, fuzzy, iterative learning, etc., methods) for updating the design parameters of LNTCS. Such methodologies, render many advantages contrary to the simulation/theoretical-based techniques. AFT belongs to the family of the so-called AO methods, such as the SPSA algorithm presented in Section 3.4.4. These methods, provide probably the most promising approach for the development of a systematic methodology for automatic, safe, robust and efficient maintenance and renovation of LNTCSs. The basic functioning procedure for AO methods may be summarized as follows (Figure 4.1):

- The traffic flow process (e.g. urban road network) is controlled in real time by a control strategy (of any kind) which includes a number of parameters to fine-tune.
- At the end of appropriately defined periods (e.g. at the end of each day), the AFT algorithm receives the value of the real (measured) performance index (e.g. average speed over space and time for traffic networks, total number of containers loaded/unloaded for seaport container terminals, etc.), as well as some aggregated values of the most significant external factors (e.g. demand). Note that the performance index $J(\theta, x)$ is a (generally unknown) function of the external factors x and the tunable parameters to be adjusted θ.
- Using the measured quantities (the number of which increases iteration by iteration),

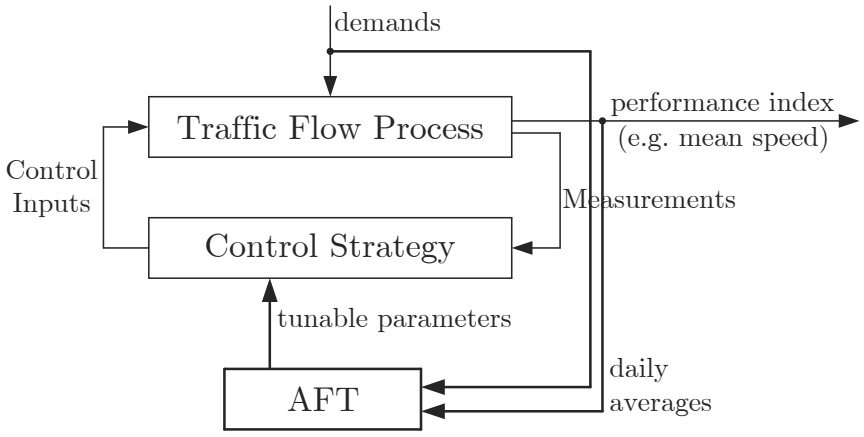

Figure 4.1: Working principle of AO for automatic calibration of LNTCSs.

the AFT algorithm calculates new tunable parameter values to be applied at the next period (e.g. the next day) in an attempt to improve the system performance.

- This (iterative) procedure is continued over many periods (e.g. days) until a maximum in performance is reached; then, the AFT algorithm may remain active for continuous adaptation or can be switched off and re-activated at a later stage (e.g. after few months).

A key idea behind most AO methodologies is to use two different (but inter-woven) phases of tunable parameter changes as follows:

1. At the **perturbation phase**, the performance of the LNTCS is evaluated at one or more random perturbations of the current set of tunable parameters.

2. At the **gradient-descent-like phase**, the current tunable parameter values are modified in a targeted way, so as to increase performance, based on an estimate of the LNTCS performance gradient. The gradient can be calculated using the values of the performance index (and of the external factors) at the perturbation phase.

The random perturbations are introduced in the perturbation phase, in order for the AO mechanisms to sufficiently explore the overall LNTCS state-space (so as to be able to come up with a suitable decision each time). As it was shown in several research articles evaluating AO methods, the introduction of random perturbations is necessary and crucial for the successful operation of the overall scheme ([42], [40], [69]). Different researchers, have reported very encouraging results – by using simulation experiments – on the application of the aforementioned AO methods in maintenance and renovation of various LNTCSs. Urban signal traffic control [70], [9], air traffic management [33], vessel traffic management [8] and

fleet and transit management [27], [34] are some of the LNTCS maintenance applications where these methods have been applied and evaluated through extensive simulation studies.

Unfortunately, these designs suffer from two severe drawbacks:

1. The majority of AO methods do not have any mechanism to incorporate the knowledge captured in the past, regarding the dependence of the LNTCS performance on the tunable parameters and the external factors (demand). In cases where such a dependence is highly nonlinear and complex, the aforementioned algorithms fail to produce any improvement of the overall LNTCS performance.

2. Most importantly, the use of random perturbations in the AO algorithms may lead to an unacceptable value of the LNTCS performance; even a small perturbation of a "good" set of tunable parameters may lead to an unacceptable or, even worse, unstable or catastrophic behavior. Hence, AO methods possess the disadvantage of not guaranteeing efficient and, most importantly, safe performance during the perturbation phase.

The purpose of this thesis is to introduce and analyze a new AO algorithm which is capable of overcoming the limitations (1)–(2) above. The developed approach appropriately combines the nice features of AO algorithms with those of approximation theory and adaptive mechanisms. The resultant adaptive optimization methodology is capable of rapidly and efficiently optimizing systems of arbitrary complexity and scale, such as LNTCS and, most importantly, guaranteeing robust and safe performance while the maintenance operation is on.

The main components of the employed algorithm are summarized as follows:

- An approximator $\widehat{J}(\theta, x)$ is used (e.g., a neural network or a polynomial-like approximator) in order to obtain an approximation of the nonlinear mapping $\widehat{J}(\theta, x) = J(\theta, x)$.

- An on-line adaptive/learning mechanism is employed for "training" the above approximator. Globally convergent learning algorithms (see e.g., [38], [36]) are required for such a purpose.

- At each algorithm iteration k, many randomly chosen candidate perturbations (of dimension n_θ) of vector $\theta(k)$ are generated. The effect of each of these perturbations to the LNTCS performance is estimated by use of the approximator mentioned above. The perturbation that corresponds to the "best" estimate (i.e., the one that leads to the best value for $\widehat{J}$) is picked to depict the new values for the tunable parameters $\theta(k+1)$. That is, $\theta(k+1)$ corresponds to the best estimate of $\widehat{J}$, selected to be applied at the next period (e.g. the next day).

4.5 Universal approximators

The proposed adaptive/learning algorithm, presented in next section, makes use of a so called universal approximator, which has the form $\widehat{J}(\theta, x, \vartheta)$, with ϑ denoting the vector of the approximator's tunable parameters. Universal approximators have very strong functional capabilities, that is, they have the ability to approximate broad classes of functions to within any level of accuracy. If properly constructed, they can perform very complex operations and implement very complex nonlinear mappings (e.g., much more complex than those that can be implemented by a linear mapping).

To study this idea, let $\Im$ denote the set of all possible approximator structures of type $\widehat{J}(\theta, x, \vartheta)$. For example, if $\widehat{J}(\theta, x, \vartheta)$ is a polynomial-like approximator, then $\Im$ would contain an infinite number of approximator structures, each one with different polynomials (nonlinear activation functions) and different values for the approximator parameters ϑ. If $J(\theta, x)$ is any real valued, bounded, continuous function, and for an arbitrary $e > 0$ there exists an approximator structure $\widehat{J}(\theta, x, \vartheta) \in \Im$ such that

$$\sup_{x \in X, \theta \in \Theta} \left| J(\theta, x) - \widehat{J}(\theta, x, \vartheta) \right| < e \tag{4.8}$$

then the approximator structure $\widehat{J}(\theta, x, \vartheta)$ is said to satisfy the "universal approximation property". Actually, radial basis function neural networks and standard and Takagi-Sugeno fuzzy systems are known to satisfy the "universal approximation property" [62]. Clearly, however, the linear approximator structures do not satisfy the universal approximation property.

An approximator that satisfies the universal approximation property is flexible enough to be able to represent many functions; however, this "flexibility" may require a large approximator structure and extraordinary parameter tuning capabilities.

Satisfaction of the universal approximation property guarantees that there exists a way to define the particular approximator structure $\widehat{J}(\theta, x, \vartheta)$ and its parameters, in order to represent the unknown nonlinearity as accurately as you would like. It does not, however, say how to find the particular $\widehat{J}(\theta, x, \vartheta)$, which can, in general, be very difficult (i.e., it does not say how many neurons should be in the neural network, how many rules should be in the fuzzy system, or anything about how to pick the approximator's parameters in order to get good accuracy). Furthermore, for arbitrary accuracy you may need an arbitrarily large number of parameters (i.e., you may need to include an arbitrarily large number of regressor terms in the approximator structure).

The value of the universal approximation property is simply that it shows, that if you work hard enough at structure choice, are willing to use a large structure and do good parameter tuning, you should be able to make the neural networks and fuzzy systems achieve what you are trying to get done. For estimation, this means that there is great flexibility in working with the neural and fuzzy approximator structures. If you choose enough neurons or enough rules and membership functions in a fuzzy system, there is a way to tune the neural or fuzzy system so that it will perform its estimation task very well. For control, practically speaking, it means that there is great flexibility in tuning the nonlinear function implemented by, e.g., the fuzzy system or a neural network.

To summarize, for a given approximator structure $\widehat{J}(\theta, x, \vartheta)$ that satisfies the universal approximation property, all we know is that there exist a bound $W > 0$ on the representation error of the unknown function $J(\theta, x)$ with the approximator $\widehat{J}(\theta, x, \vartheta)$. We typically do not know how small it is. The universal approximation property simply says that we may increase the size of the approximation structure and properly define the parameters of the approximator to achieve any desired accuracy (i.e., to make W as small as we want); it does not say how big the approximator must be or if you fix the structure $\widehat{J}(\theta, x, \vartheta)$ how small W is. In the following subsections two different universal approximators are presented. Both of them where used within the AFT algorithm in order to approximate the performance index of the control system under study.

4.5.1 Linear-in-the-weights Universal Approximator (LUA)

One of the universal approximators used in this thesis' experiments in order to approximate the objective function $J(\theta, x)$, is a linear-in-the-weights polynomial-like approximator with L_g regressor terms, which takes the form

$$\widehat{J}(\theta, x) = \vartheta^T \phi(\theta, x) \tag{4.9}$$

where ϑ denotes the vector of the approximator parameter estimates and

$$\phi(\theta, x) = \left[\phi_1(\theta, x), \phi_2(\theta, x), \ldots, \phi_{L_g}(\theta, x)\right]^T. \tag{4.10}$$

The non-linear functions $\phi_i(\theta, x)$ are given by

$$\phi_i(\theta, x) = S^{d_1}(\theta_{m_1}) \cdot \bar{S}^{d_2}(x_{m_2}) \cdot S^{d_3}(\theta_{m_3}),\ d_i \in \{0, 1\} \tag{4.11}$$

where d_i, m_i are randomly chosen at each iteration of the AFT algorithm (with $m_1, m_3 \in \{1, 2, \ldots, n_\theta\}$, $m_2 \in \{1, 2, \ldots, n_x\}$ and $\sum_i m_i \in \{2, 3\}$) and $S(\cdot), \bar{S}(\cdot)$ are smooth monotone nonlinear functions. In the neural networks literature [49], [28] these functions are usually

chosen to be "sigmoidal". In the simulation applications presented in Chapters 5, 6 we choose

$$S(\theta) = \tanh(\lambda_1 \theta + \lambda_2), \quad \bar{S}(x) = \tanh(\lambda_3 x + \lambda_4) \tag{4.12}$$

where λ_i are non-negative real numbers initially defined by the user; after 4–5 iterations of the algorithm the values of λ_i are optimized so as to minimize

$$\min \sum_{\ell=1}^{k-1} \left(J_\ell - \vartheta^T \phi_\ell^{(k)} \right)^2. \tag{4.13}$$

The factors ℓ_i are relevant to the normalization of the values of the approximator inputs θ, x.

4.5.2 Adaptive Neuro-Fuzzy Inference System (ANFIS)

The last decades, fuzzy logic is commonly used in control systems as it provides an easy way to deal with possible system uncertainties and noisy data. By the use of **membership functions** and if-then **rules** one can form a fuzzy estimate system in order to perform a nonlinear input-output mapping. Fuzzy inference systems belong to the family of universal approximators, as they can accurately estimate broad classes of nonlinear functions to within any level of accuracy. The most famous fuzzy inference systems are Mamdami-type and Sugeno-type [50], [75], [74].

The second universal approximator that was investigated in this thesis' experiments is the MATLAB's **toolbox** ANFIS (abbreviation for Adaptive Neuro-Fuzzy Inference System). In such neuro-fuzzy systems, the theory of neural networks is combined to the nice features of fuzzy logic in order to provide better performance and powerful approximation capabilities. In our case, a Sugeno-type FIS with L_g regressor terms is selected, which takes the form

$$\widehat{J}(\theta, x) = \vartheta^T \phi(\theta, x) \tag{4.14}$$

where ϑ denotes the vector of the FIS adjustable parameters and

$$\phi(\theta, x) = \left[\phi_1(\theta, x), \phi_2(\theta, x), \ldots, \phi_{L_g}(\theta, x) \right]^T. \tag{4.15}$$

The non-linear functions $\phi_i(\theta, x)$ correspond to membership functions of arbitrary distributions. Sugeno-type FIS are appropriate for systems that have many inputs and one scalar output (like the one we consider in our problem). ANFIS combines the advantages of neural networks and fuzzy logic in order to fit the input-output data to the created model, usually using a trial and error method.

The user can choose among a wide variety of distributions for the memberships functions $\phi_i(\theta, x)$. Sugeno-type FIS uses a hybrid learning algorithm [30], that iteratively changes the values of the adjustable parameters until the fitting error between the data and the

modeled system in minimized. The learning algorithm combines least squares estimation and backpropagation method in order to optimally adjust the membership functions' parameters ϑ. The iterative process uses a first forward pass (which propagates the data) and a second pass where the least squares method calculates the approximation error. The calculated error is then backward to the adjustable parameters using a gradient descent update method.

The aforementioned learning algorithm along with the experience of the user, provide ANFIS approximators with flexibility and powerful approximation advantages. Thus, they are appropriate for application to systems like these considered in Chapters 5 and 6.

4.5.3 Bias-variance tradeoff

In the previous subsections two different universal approximators were presented. This section discusses the problem of how can one best use data to select an appropriate approximator about a system (again, this section is widely based on Spall's book [72]). Choosing a model for the system approximation is a stochastic search and optimization process itself ("find the best model"), as no model is "perfect". The fundamental tradeoff between the bias and variance, which is discussed in details below, could be used in order to choose a model form. This tradeoff, provides a structure for balancing the need to have a relatively simple model that is easy to interpret and the need to have a model sufficiently rich to capture all relevant linear or nonlinear effects.

The **bias-variance tradeoff** is a fundamental principle in comparing the quality of different mathematical models. In Chapters 3 and 4, we have assumed the existence of a loss function $J(\theta)$ and/or a root-finding function $g(\theta)$ although they may not be directly available, corresponding to cases with noisy function measurements. We then described methods by which θ can be "optimized". A closely related issue, however, is determining the mathematical **form** of the functions $J(\theta)$ or $g(\theta)$ **prior** to optimizing θ. In the linear or nonlinear regression context, this is essentially tantamount to determining the form of an underlying approximation model describing the input-output relationship of the system of interest. A typical problem to solve, is the number of terms that should be included in a polynomial approximator of an unknown function of interest.

One of the most common problems in approximation methodologies is that of over- and under-fitting. The bias-variance tradeoff provides a formal structure, which states, in essence, that one should seek simpler models over more complex models. Of course, to achieve optimal predictive power from a mathematical model, it is also necessary to include sufficient richness in the model to capture the essential characteristics of the process. Hence, one should not use a model that is **too simple**. This formal structure, however, does not lead directly to implementable algorithms for realizing an optimal tradeoff between the bias and the variance.

That will await the next section, which considers into the cross-validation method for model selection.

Let $J(\theta, x)$ represent the scalar output for some system based on input vectors θ, x. Suppose, as in previous sections, that we model this input-output process with a regression function $\widehat{J}(\theta, x)$ and a noise term. In particular, the model for the actual output $J(\theta, x)$ is the right-hand side of

$$J(\theta, x) = \widehat{J}(\theta, x) + e \tag{4.16}$$

where e is a noise term that may or may not have mean zero. The model on the right-hand side above does not generally correspond to the actual mechanism for generating the true output $J(\theta, x)$.

A natural measure of effectiveness of the regression function $\widehat{J}(\theta, x)$ as a predictor of $J(\theta, x)$, given the current values of vectors θ and x, is the mean-squared error (MSE), $\left[J(\theta, x) - \widehat{J}(\theta, x)\right]^2$, which is directly related to the used model. It is for this model error that we are interested in a bias-variance analysis.

An analysis of the squared error of the model $\left[J(\theta, x) - \widehat{J}(\theta, x)\right]^2$ provides direct insight into the quality of the regression function $\widehat{J}(\theta, x)$ as a predictor of $J(\theta, x)$, given θ, x. In order to estimate the real function $J(\theta, x)$ we need different sets of input-output data. Hence, we must average the squared error over possible values of θ and x, which reflect the input-output data.

Let $\{(\theta_1, x_1, J_1), (\theta_2, x_2, J_2), \ldots, (\theta_n, x_n, J_n)\}$ denote n input-output data that will be collected to form the estimate of $J(\theta, x)$, based on an appropriate search and optimization algorithm. A useful approach to analyzing the inherent model quality is to take the mean of the squared model error $\left[J(\theta, x) - \widehat{J}(\theta, x)\right]^2$, based on averaging with respect to the **distribution for the fitting data**. This is equivalent to averaging with respect to the resulting distribution for $\widehat{J}(\theta, x)$. It can be shown (see [72] for details) that for any future θ_{n+1}, x_{n+1} this MSE is

$$\begin{aligned} \text{MSE} &= \left[\widehat{J}(\theta_{n+1}, x_{n+1}) - \bar{J}(\theta_n, x_n)\right]^2 + \left[J(\theta_{n+1}, x_{n+1}) - \widehat{J}(\theta_{n+1}, x_{n+1})\right]^2 \\ &= (\text{variance at } \theta_{n+1}, x_{n+1}) + (\text{bias at } \theta_{n+1}, x_{n+1})^2 \end{aligned} \tag{4.17}$$

where $\bar{J}(\theta_n, x_n)$ the average of $\widehat{J}(\theta, x)$ over the first n data sets. An unbiased estimator is one with $\widehat{J}(\theta_{n+1}, x_{n+1}) = J(\theta_{n+1}, x_{n+1})$ (implying that the second term on the right-hand side of (4.17) is zero).

As a final, **overall** assessment of contributions toward the model MSE, one can average the squared bias and variance in (4.17) over all possible values of (θ, x), yielding mean values, say $\overline{\text{bias}^2}$ and $\overline{\text{variance}}$. Averaging the variance and squared bias terms in (4.17) leads to a

global measure of the contributions to the MSE that does not depend on a specific set of values of vectors (θ, x):

$$\begin{aligned} \text{MSE}_{\text{overall}} &= E_{\theta,x}\left[\text{variance at } \theta, x + (\text{bias at } \theta, x)^2\right] \\ &= \overline{\text{variance}} + \overline{\text{bias}^2} \end{aligned} \tag{4.18}$$

where $E_{\theta,x}[\cdot]$ denotes the appropriate average over possible values of (θ, x).

Although unbiasedness is generally considered a desirable property, (4.17) and (4.18) show that an unbiased estimator can have a large $\text{MSE}_{\text{overall}}$ when the variance is large. In particular, a biased estimator may have a lower $\text{MSE}_{\text{overall}}$ than an unbiased one, even better than unbiased estimators that are best by some criterion (such as an estimator with the lowest variance among unbiased estimators that are linear combinations of the measurements). There is generally a tradeoff between the variance and bias contributions to the overall MSE. Regression functions $\widehat{J}(\cdot)$ with high variance tend to have low bias and vice versa. More generally, when $\widehat{J}(\cdot)$ depends on inputs (θ, x), there is a relationship between the complexity of the model and the relative bias and variance. In particular, the following relationships typically hold:

Simple model	$\Longleftrightarrow$	**High bias/low variance**
Complex model	$\Longleftrightarrow$	**Low bias/high variance**

The bias-variance tradeoff provides a framework for choosing among candidate models. Of course, in practice, many factors other than bias, variance, and the resulting $\text{MSE}_{\text{overall}}$ may be relevant. These include cost, development time, historical precedent for particular model forms, desires of organizational leadership, and so on. Nevertheless, all other factors being equal, one would wish to pick the function $\widehat{J}(\cdot)$ with a balanced bias and variance. This balance in bias and variance results in the minimum overall MSE according to (4.17) and (4.18).

Unfortunately, the bias-variance tradeoff is largely limited to gaining a **conceptual** understanding for comparing different models. Because the probability distributions for the input-output data are not known (they depend on knowing the unknown true model), the values of the bias and variance will generally be unknown. It is, however, clear from the bias-variance tradeoff that there can be no universal best model form. One model form may provide nicely balanced bias and variance on one class of problems, but be too rigid (high bias) or flexible (high variance) in another example. For a **fixed** model form, the variance contribution to MSE tends to decrease when the sample size used in fitting the model is

increased. Intuitively, this follows since the model quality improves from the greater information available for fitting the model. The variance contribution to MSE decreases with a greater amount of data, since there is a reduced tendency to fit to the individual data points.

4.5.4 Cross-validation for model selection

There are a great number of methods for approximately addressing the bias-variance tradeoff in a manner that is feasible for implementation. These methods are variations on the theme of balancing low- and high-order requirements to produce a model that (implicitly at least) balances the bias and variance. This section focuses on cross-validation [73], one of the most popular and flexible means of realizing an optimal tradeoff between bias and variance (according to Spall's book [72]). Cross-validation provides a mechanism for choosing between candidate model forms and it is widely applicable. The basic principle in model selection is to minimize, implicitly or explicitly, a criterion of the generic form

$$f_1(\text{fitting error from given data}) + f_2(\text{model complexity}), \tag{4.19}$$

where f_1 and f_2 are increasing functions of, respectively, some measure of the error in the model predicting the fitting data (the fitting error) and some measure of the number of terms in the model (model complexity).

The cross-validation approach is perhaps the most straightforward formal model selection method to understand and to implement. It is based on manipulations of the fitting (training) data (i.e., the data assumed available for model estimation). Cross-validation does not require additional data and/or detailed prior information or detailed analysis beyond sample model fits. It also has the advantage of applying to candidate models of virtually any form, not being restricted to specific classes of candidate models (e.g., linear/curvilinear regression models), as other approaches. Finally, unlike some other approaches, it does not require that the underlying data be normally distributed. On the other hand, cross-validation is not necessarily the most powerful or discerning method in any specific problem, nor is it the most computationally efficient (since it requires repeated "sample" model fits). Cross-validation is one of the model-selection methods that **implicitly** optimize the tradeoff criterion (4.19), as there is no direct construction of a performance metric dependent on f_1 and f_2.

This section provides just a short description of how cross-validation works in the context of selecting a model. Suppose that two or more candidate model forms are to be evaluated. For example, one may have small-, medium-, and large-scale neural networks as candidate model forms and wishes to know which neural network is likely to produce the best predictions. Cross-validation is a commonsense approach based on sequentially partitioning the full data set into fitting and test subsets. For each partition, estimates are produced for the

candidate model forms from the fitting subset. Then, the performance of each candidate model is measured on the test subset. This procedure is repeated for all partitions of the full data set.

Let n_T denote the size of the test subsets, where, of course, $n_T < n$, with n the size of the full data set. A common strategy called **leave-one-out** is to pick $n_T = 1$ and cycle through all n possible combinations of fitting and test subsets (e.g., [73]). This approach produces n model fits from the n possible fitting subsets of size $n-1$. Each of these model fits generates a prediction error on the one data point left out (i.e., the difference between the outcome of the point left out and the predicted value based on the model fit from the remaining $n-1$ points). The best model form is the one for which the chosen type of average for these n prediction errors – say, the sample MSE or mean absolute deviation (MAD) – is lowest.

There are often advantages, however, to choosing $n_T > 1$. The advantages include greater efficiency (i.e., fewer model fits) for some implementations with $n_T > 1$. When $n_T > 1$, the test subsets may be chosen deterministically or randomly, with or without replacement. For test subsets chosen deterministically with replacement (i.e., all possible combinations of test subsets of size n_T are used), there are a potentially huge number of possible fitting/test subset combinations. In particular, the number of combinations is "n choose n_T" $\binom{n}{n_T}$.

One way of mitigating this explosion is to randomly select (usually with replacement) a relatively small number of test subsets of size n_T (e.g., [68]). Another approach, is to choose n_T such that n is divisible by n_T and then choose the test subsets **without replacement**, so that all of the data appear once and only once in a test subset. The allocation of the n data to the n/n_T test subsets may be done randomly or deterministically. The "once and only once" aspect may be viewed as an extension of the leave-one-out strategy. This allocation reduces the number of model fits from n in leave-one-out and from $\binom{n}{n_T}$ in deterministic replacement selection to n/n_T.

4.6 The structure of AFT algorithm

This section presents the AFT algorithm. The general concept of this method is directly connected to RDSA. Practically, the intuition of AFT can be considered as a combination of RDSA and SPSA algorithms, as is makes use of many random perturbations of the tunable parameters θ at every algorithm iteration k; however, it does not require to run the simulation experiments, as the random vectors are evaluated by the universal approximator $\widehat{J}(\theta, x)$.

Table 4.1: Variables used within the AFT.

k	iteration index
ℓ	past performance measurements index
$J(\ell)$	performance value for the ℓ-th calibration experiment
$\widehat{J}(\ell)$	an estimate of $J(\ell)$ obtained at the ℓ-th iteration
$\theta(k)$	the vector of tunable parameters at the k-th calibration experiment
$\theta^*(k)$	the "best" set of tunable parameters until the k-th experiment
$x(k)$	the exogenous signals as defined in Section 4.2
$\bar{x}(k)$	an estimate/prediction of the exogenous signals $x(k)$
$\Delta_i(k)$	zero-mean random sequences (e.g. Gaussian), with $i \in \{1, 2, ..., n_\theta\}$
$\Delta\theta(k+1)$	the perturbation picked by the algorithm

4.6.1 Basic notations

Table 4.1 presents a description of the design parameters and variables used within the AFT algorithm. For the implementation of AFT, it is assumed that an estimate, or prediction, $\bar{x}(k+1)$ of the vector $x(k+1)$ is available (referring to the estimate of next iteration). In many applications such an assumption is quite realistic, since the entries of $x(k+1)$ correspond to system states and exogenous inputs, which are available or measurable. However, there may be cases where such an assumption is not realistic; in this case $\bar{x}(k+1)$ can be estimated/predicted using appropriate estimation algorithms. It should be noted, that the dimension n_x of the estimate vector $\bar{x}$ should be comparative to the dimension n_θ of the vector of tunable parameters θ.

Contrary to other applications of neural approximators, where the number of neurons $\bar{L}_g$ should be large enough to guarantee efficient approximation over the whole input set, this is not the case here. In the case of the developed algorithm, it is sufficient, that the approximator $\widehat{J}$ has enough regressor terms to come up with an approximation of the unknown function J over a "small neighborhood", around different sets of values of vector θ.

4.6.2 Algorithm description

Table 4.2 presents a step-by-step description of AFT algorithm application. It is worth noting, that similarly to RDSA, the proposed algorithm introduces random perturbations to the control design parameter vector θ. Besides, the use of random perturbations is crucial for the efficiency of the proposed algorithm as it provides the so-called persistence of excitation

Table 4.2: AFT algorithm mathematical description.

Step 1:	$\Delta\theta_i^{(j)}(k+1) = \alpha(k)\Delta_i^{(j)}(k) - \theta_i(k) + \theta_i^*(k),\ j \in \{1,\dots,K\}$
Step 2:	$L_g(k) = \min\left\{2\left(k-1\right), \bar{L}_g\right\}$
Step 3:	$\ell(k) = \max\left\{k - T_h, 1\right\}$
Step 4:	$\phi_\ell(k) = \phi\left(\theta_\ell(k), \bar{x}_\ell(k)\right)$
Step 5:	$\vartheta(k) \mapsto \arg\min_\vartheta \frac{1}{2}\sum_{\ell=\ell(k)}^{k}\left(J_\ell - \vartheta^\tau\phi_\ell(k)\right)^2$
Step 6:	$\widehat{J}\left(\pm\Delta\theta^{(j)}(k+1) + \theta^*(k), \bar{x}(k+1)\right) = \vartheta(k)^\tau\phi\left(\pm\Delta\theta^{(j)}(k+1) + \theta^*(k), \bar{x}(k+1)\right)$
Step 7:	$\Delta\theta(k+1) = \arg\max_{\Delta\theta^{(\pm j)}(k+1)} \widehat{J}\left(\pm\Delta\theta^{(j)}(k+1) + \theta^*(k), \bar{x}(k+1)\right)$
$\Delta\theta_i(k+1) = \theta_i(k+1) - \theta_i(k)$	
$\alpha(k)$ is a user-defined positive sequence (e.g. constant stepsize $\alpha(k) \equiv \alpha \in (0,1)$)	
$T_h, \bar{L}_g, K$ are user-defined positive integers	
$\theta^*(k) + \Delta\theta(k+1)$ denotes the vector of tunable parameters picked to be applied at the next experiment $k+1$	

(PE) property, which is a sufficient and necessary condition for the neural approximator $\widehat{J}$ to be able to efficiently learn the unknown function J. However, due to the use of Step 6 (see Table 4.2) the proposed methodology avoids poor performance or instability problems, and guarantees safe and efficient performance in the sense that requirement (4.7) is fulfilled.

Below, we discuss in details the application steps of the algorithm displayed in Table 4.2, which are performed at every iteration k:

- **Step 1:** *Calculate K random perturbations.* In this step K random perturbations are calculated (according to e.g. Gaussian distribution). The resulting candidate vectors $\theta_i(k+1) = \theta_i^*(k) + \Delta\theta_i(k+1)$ are then projected in Θ, in order to satisfy the problem constraints.

- **Step 2:** *Calculate the number of approximator regressor terms.* The number of the approximator's regressor terms $L_g(k)$ to be used for this iteration is calculated.

- **Step 3:** *Calculate the number of past measurements.* The algorithm keeps a window of past measurements which moves along with the iterations. In this step the starting point of the window in the past is calculated. The end point of the window is always

k.

- **Step 4:** *Produce the polynomial-like approximator.* After steps 2, 3 the structure of the universal approximator may be formed and applied to the window of the past measurements.

- **Step 5:** *Calculate the optimal approximator parameter estimates.* The optimal values of the approximetor's parameters ϑ are calculated according to the solution of a least squares estimation method.

- **Step 6:** *Apply the* $2K$ *random perturbations* $\pm\Delta\theta(k+1)^{(j)}$ *to the* $\widehat{J}(k)$. The $2K$ candidate vectors $\theta^*(k)\pm\Delta\theta(k+1)^{(j)}$ are applied to the approximator $\widehat{J}(k)$ for evaluation.

- **Step 7:** *Pick the "best" random perturbation (according to the* $\widehat{J}_k$*).* The vector $\theta(k+1)$ with the best estimated performance is selected for application to the next simulation experiment.

As shown in [39], [40], [41], [42] using strict mathematical arguments, if the structure of the approximator and its learning mechanism satisfy certain design considerations (that are independent of the particular application), then the above described process guarantees rapid convergence of the overall maintenance procedure (in the sense of (4.6)) to the same performance levels that would have been obtained if efficient non-linear optimization schemes, such as the steepest descent or Gauss-Newton schemes, could be applied to the particular problem. Most importantly, the above-mentioned procedure guarantees safe, stable and efficient transient performance in the sense that the system performance during maintenance remains within acceptable levels (satisfying inequality (4.7)). This performance, can be, in the worst case, similar to the system performance before maintenance has started plus some random term. The magnitude of this term is proportional to the magnitude and variance of the exogenous inputs (see equation (4.7)).

4.7 Choice of stepsizes for SA methods

The choice of the gain sequence $\alpha(k)$ is critical for the performance of SA methods. In many applications, a constant stepsize is used (instead of a descending one) as a way of avoiding gains that are too small for large k. Typical applications with constant stepsizes involve adaptive tracking or control problems where θ^* is changing in time. The constant gain provides enough impetus to the algorithm to keep up with the variation in θ^*, in contrast to a decaying gain, which provides too little weight to the current input information to allow

for the algorithm to track the solution. Such constant-gain algorithms are also frequently used in neural network training even when there is no variation in the underlying θ^* ([44]).

On the other hand, there is considerable appeal to the idea that the stepsize should depend on the actual trajectory of the algorithm. When the stepsizes depend on the observations we say that we are using a **stochastic stepsize**. Assume that our estimates are consistently under or consistently over the actual observations. This can easily happen during early iterations due to either a poor initial starting point or the use of biased estimates, which is common during the early iterations. For large-scale problems, it is possible that we have to estimate thousands of parameters. It seems unlikely that all the parameters will approach their optimal value at the same rate (wide variation in learning rates can occur). Stochastic stepsizes try to adjust to the data in a way that keeps the stepsize larger while the parameter being estimated is still changing quickly. Balancing noise against the change in the underlying signal, particularly when the noise is unknown, is a difficult challenge.

Another common ad hoc "trick" to avoid potentially sluggish behavior is to periodically restart the algorithm. That is, after starting the search with an initial condition $\theta(0), \alpha(0)$, periodically restart the gain sequence at $\alpha(0)$. Finally, there exist a number of stochastic analogues of the Newton-Raphson search in the context of parameter estimation for **particular** (possibly linear) models. The scalar gain $\alpha(k)$ is then replaced by a matrix that approximates the (unknown) true inverse of the Jacobian (Hessian) matrix. While these methods provide effective ways to increase convergence speed for special cases, they are restricted in their range of application.

4.7.1 Convergence properties

The theory for proving convergence of stochastic gradient algorithms was first developed in the early 1950s and has matured considerably since then [63]. However, all the existing proofs require three basic conditions about the applied stepsizes

$$\alpha_k \geq 0, \quad k = 0, 1, ..., \tag{4.20}$$

$$\sum_{k=0}^{\infty} \alpha_k = \infty, \tag{4.21}$$

$$\sum_{k=0}^{\infty} (\alpha_k)^2 < \infty. \tag{4.22}$$

Equation (4.20) obviously requires that the stepsizes be nonnegative. The most important requirement is (4.21), which states that the infinite sum of stepsizes must be infinite. If this condition did not hold, the algorithm might stall prematurely. Finally, condition (4.22)

requires that the infinite sum of the squares of the stepsizes be finite. This condition, in effect, requires that the stepsize sequence converge "reasonably quickly". A good intuitive justification for this condition is that it guarantees that the variance of our estimate of the optimal solution goes to zero in the limit.

All three above mentioned conditions provide a careful balance in having the gain $\alpha(k)$ decay neither too fast nor too slow. In particular, the gain should approach zero sufficiently fast to damp out the noise effects as the iterate gets near the solution θ^*. On the other hand, it should approach zero at a sufficiently slow rate to avoid premature (false) convergence of the considered algorithm. All existing proof techniques lead to these three requirements on the stepsize. Fifty years of research in this area has not been able to relax them.

Arithmetic sequences such as

$$\alpha(k) = \frac{1}{k+1}, \quad \alpha(k) = \frac{\alpha(0)}{\alpha(0)+k}, \quad \alpha(k) = \frac{1}{(k+1)^n}, \quad k = 0, 1, \ldots,$$

satisfy these three conditions. Also, there are several famous formulas about calculating descending stepsize sequences. Some of them are:
McClain's formula:

$$\alpha(k) = \frac{\alpha(k-1)}{1+\alpha(k-1)-\bar{a}}$$

Kesten's formula:

$$\alpha(k) = \frac{\alpha(0)}{\alpha(0)+K^k-1}$$

Belgacem's formula:

$$\alpha(k) = \frac{1}{K^{k+1}}$$

with $\alpha(0)$, $\bar{a}$, K being some design parameters. According to the special practical problem one should try different descending rules to find one that is suitable for the specific application. Figure 4.2 presents the descending sequence $\alpha(k) = \alpha(0)/\left(\alpha(0)+k\right)$ for $k = 1, 2, \ldots, 50$ and different values for parameter $\alpha(0)$.

4.7.2 An adaptive technique for the stepsize calculation

For our experiments, we introduce an adaptive technique for the calculation of stepsize $\alpha(k)$, at each iteration of AFT algorithm k. This technique is based on the signs $(\pm)$ of the differences $\Delta\theta(k)$, $\Delta\theta(k-1)$ of the last two iterations. If there are frequent sign changes, this is an indication that the iterate is near θ^*; if the signs are not changing, this is an indication that the iterate is far from θ^*. This forms the basis for an adaptive choice of the gain $\alpha(k)$, where a larger gain is used if there are no sign changes and a smaller gain is used if the signs change frequently.

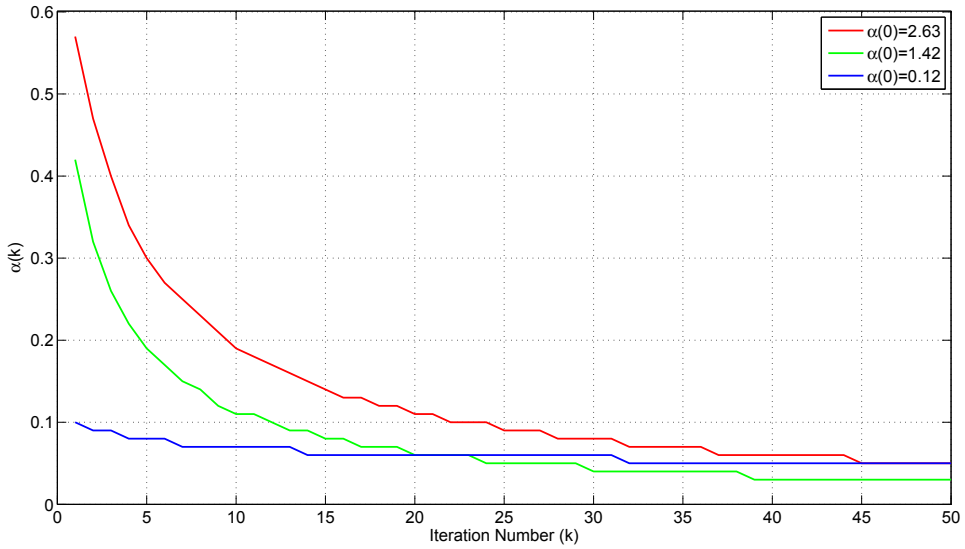

Figure 4.2: A famous descending sequence for stochastic stepsizes $\alpha(k)$.

Given the desirability for a gain sequence that balances algorithm stability in the early iterations with non-negligible gains in the later iterations, a proposed stepsize form is:

$$\alpha(k) = \frac{\alpha(0)}{\alpha(0) + K_i}. \tag{4.23}$$

Initially $K_i = 1, \forall i$ and then for every iteration $k = 2, 3, \ldots$ we have:

$$K_i = \begin{cases} K_i & \text{if } \Delta\theta_i(k)\Delta\theta_i(k-1) > 0 \\ K_i + 1 & \text{if } \Delta\theta_i(k)\Delta\theta_i(k-1) < 0. \end{cases} \tag{4.24}$$

This form is inspired by the famous learning method RPROP [64] and the arithmetic sequences mentioned in previous Subsection. This formula takes into account only the sign of $\Delta\theta_i$ and acts independently on each θ_i. This way, every element of vector θ converges with a different rate according to the frequency of sign changes.

4.8 Other requirements for convergence

AFT is a direct analogue of RDSA and SPSA algorithms. Although AFT uses a universal approximator to estimate the outputs of function $J(\theta, x)$ it can be easily shown that it requires the same properties for the underlying system in order to prove convergence. Proves for RDSA and SPSA can be found in many places in the literature (e.g. [72]) both from the

"statistical" and the "engineering" point of view. The conditions for an almost sure (a.s.) convergence are the following:

- The three conditions defined for the stepsize in Section 4.7.1.
- Unique minimum for θ^*.
- Zero-mean and finite variance for the noise in the measurements.
- Smoothness of $J(\theta, x)$. J should be two-times continuously differentiable and bounded for all $\theta \in \mathrm{R}^{n_\theta}$.
- Statistical properties of the random perturbations. The $\Delta_i(k)$ have to be independent for all k, i, identically distributed for all i at each k, symmetrically distributed about zero and uniformly bounded in magnitude for all k, i.

The convergence conditions above provide an abstract ideal. In practice, one will rarely be able to check all these conditions due to a lack of knowledge about function J. In fact, the conditions may not be verifiable for the very reason that one is using such a stochastic search algorithm! Nonetheless, the conditions are important in identifying the types of problems for which there are guarantees of algorithm convergence. Also, conditions on J that may be formally unverifiable may be at least intuitively plausible, providing some sense that the algorithm is appropriate for the problem.

Chapter 5

Application to ramp metering control

This chapter presents the application results the AFT algorithm to a large-scale ramp metering problem. Section 5.1 introduces the reader to the concept of ramp metering and Section 5.2 describes some well-known existing ramp metering methodologies. In Section 5.3 the macroscopic simulator METANET which is used for the simulation experiments is presented, while in Section 5.4 a short description of the studied motorway network in Melbourne, Australia is provided. Section 5.5 discusses the application set-up of the algorithm for this particular network implementation and Section 5.6 examines and analyzes the results of the simulation experiments. Finally, Section 5.7 provides some concluding remarks about the obtained results for the two simulated control scenarios.

5.1 Introduction

Motorways had been originally conceived so as to provide virtually unlimited mobility to road users, without the annoyance of flow interruptions by traffic lights. The rapid increase of traffic demand, however, led soon to increasingly severe congestions, both **recurrent** (occurring daily during rush hours) and **nonrecurrent** (due to incidents). The increasingly congested motorways within and around metropolitan areas resemble the urban traffic networks before introduction of traffic lights (chaotic conditions at intersections, long queues, degraded infrastructure utilization, reduced safety). At the present stage, responsible authorities have not fully realized that the expensive motorway or motorway-network infrastructure is strongly underutilized on a daily basis, due to the lack of efficient and comprehensive traffic control systems. In other words, the expensive infrastructure is intended to deliver a nominal capacity that is not available (due to congestion), ironically, exactly at the time it is most urgently needed (during peak hours).

Controlling the motorway traffic flow process is a highly complicated task which may

involve a variety of spatially distributed control measures. The way the control measures behave and act on the traffic process stems from the specific design of the control strategy used. The employed control strategy determines the control actions and the specific response to the prevailing traffic conditions, through the available control actuators, based on its design and prespecified goals. The control measures that are typically employed in motorway networks are the following:

- **Ramp metering**, activated via installation of traffic lights at on-ramps or motorway interchanges.
- **Link control** that comprises a number of possibilities including lane control, variable speed limits (VSL), congestion warning, tidal (reversable) flow, keep-lane instructions, etc.
- **Driver information and guidance systems**, either by use of roadside variable message signs (VMS) or via vehicle-infrastructure integration (VII) systems with properly equipped vehicles, provide a promising technological background for efficient traffic control.

5.2 Ramp metering

Ramp metering is the most direct and efficient way to control and upgrade motorway traffic. Various positive effects are achievable if ramp metering is appropriately applied:

- increase in mainline throughput due to avoidance or reduction of congestion;
- increase in the served volume due to avoidance of blocked off-ramps or motorway interchanges;
- utilization of possible reserve capacity on parallel arterials;
- efficient incident response;
- improved traffic safety due to reduced congestion and safer merging.

Fixed-time ramp metering strategies are derived off-line for particular times of day, based on constant historical demands, without use of real-time measurements and are based on simple static models. Their basic drawback is that they may lead (due to the absence of real-time measurements) either to overload of the mainstream flow (congestion) or to underutilization of the motorway. In fact, ramp metering is an efficient but also delicate control measure. If ramp metering strategies are not accurate enough, then congestion may

not be prevented from forming, or the mainstream capacity may be underutilized (e.g., due to groundlessly strong metering). On the other hand, reactive ramp metering strategies are employed at a tactical level, i.e., in the aim of keeping the motorway traffic conditions close to prespecified set values, based on real-time measurements.

5.2.1 Local ramp metering

Local ramp metering strategies make use of traffic measurements in the vicinity of a ramp in order to calculate suitable ramp metering values (independently for each ramp). The **demand-capacity strategy** (see [51] for details), which is quite popular in North America reads

$$r(t) = \begin{cases} q_{\text{cap}} - q_{\text{in}}(t-1), & \text{if } o_{\text{out}}(t) \leq o_{\text{cr}} \\ r_{\min}, & \text{else} \end{cases} \tag{5.1}$$

where t denotes the discrete time index, q_{cap} is the motorway capacity downstream of the ramp, q_{in} is the motorway flow measurement upstream of the ramp, o_{out} is the motorway occupancy (similar to density) measurement downstream of the ramp, o_{cr} is the critical occupancy (at which the motorway flow becomes maximum, see Figure 5.1), and $r_{\min}$ is a prespecified minimum admissible ramp flow value. Figure 5.1 represents the **fundamental diagram**, which shows the relation between occupancy and flow in a motorway. The strategy (5.1) attempts to add to the last measured upstream flow $q_{\text{in}}(t-1)$ as much ramp flow $r(t)$ as necessary to reach the downstream motorway capacity q_{cap}. If, however, for some reason, the downstream measured occupancy $o_{\text{out}}(t)$ becomes overcritical (i.e. a congestion may form), the ramp flow $r(t)$ is reduced to the minimum flow $r_{\min}$ in order to avoid or to dissolve the congestion.

It is clear that the ramp flow $r(t)$ is a control input, the downstream occupancy $o_{\text{out}}(t)$ is an output, while the upstream motorway flow $q_{\text{in}}(t)$ is a disturbance. Hence, equation (5.1) does not really represent a closed-loop strategy but an open-loop disturbance-rejection policy

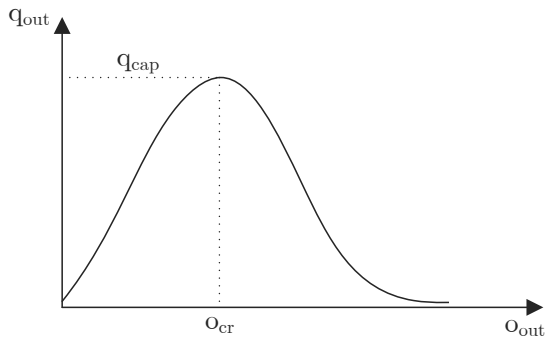

Figure 5.1: Fundamental diagram of traffic flow.

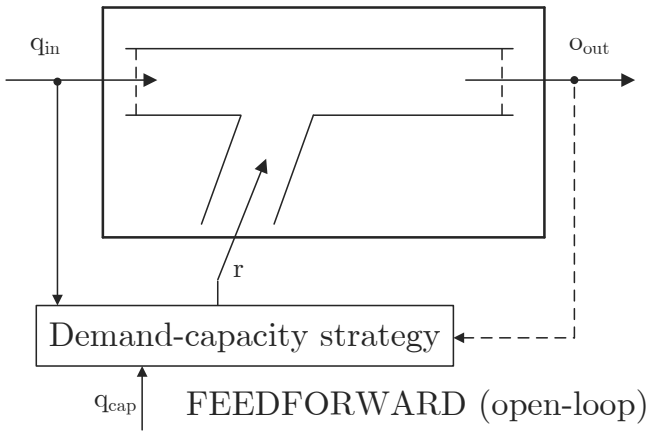

Figure 5.2: The demand-capacity local ramp metering strategy.

(Figure 5.2), which is generally known to be quite sensitive to various further nonmeasurable disturbances. The **occupancy strategy** (see [51] for details) is based on the same philosophy as the demand-capacity strategy, but it relies on occupancy-based estimation of the motorway flow upstream of the ramp q_{in}, which may, under certain conditions, reduce the corresponding implementation cost.

5.2.2 The ALINEA strategy

An alternative, local closed-loop ramp metering strategy named ALINEA was suggested in [59] and reads

$$r(t) = r(t-1) + K_{\text{R}} \left[\widehat{o} - o_{\text{out}}(t)\right] \tag{5.2}$$

where $K_{\text{R}} > 0$ is a regulator parameter and $\widehat{o}$ is a set (desired) value for the downstream occupancy (typically, but not necessarily, $\widehat{o} = o_{\text{cr}}$ may be set, in which case the downstream motorway flow becomes close to q_{cap}, see Figure 5.3). ALINEA is obviously an integral regulator, hence, it is easily seen that at a stationary state (i.e. if q_{in} is constant), $q_{\text{out}}(k) = \widehat{o}$ results from equation (5.2), although no measurements of the inflow q_{in} are explicitly used in the strategy.

Note that the demand-capacity strategy described in the previous Subsection, reacts to excessive occupancies o_{out} only after a threshold value (o_{cr}) is exceeded, and in a rather "crude" way. On the other hand, ALINEA reacts smoothly even to slight differences $\widehat{o} - o_{\text{out}}$, and thus it may prevent congestion by stabilizing the traffic flow at a high throughput level. Finally, the set value $\widehat{o}$ may be changed at any time, and thus ALINEA may be embedded into a hierarchical control system, with set values of the individual ramps being specified in real time by a superior coordination level or by an operator.

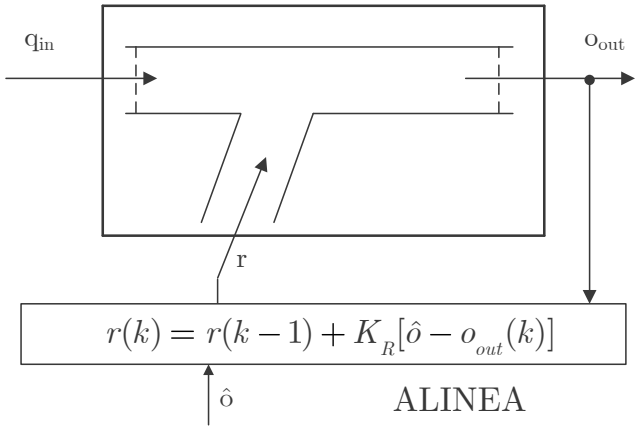

Figure 5.3: The ALINEA local ramp metering strategy.

Both aforementioned control strategies calculate suitable ramp volumes $r(t)$. In the case of traffic-cycle realization of ramp metering, $r(t)$ is converted to a green-phase duration $g(t)$ by use of

$$g(t) = \frac{r(t)}{r_{\text{sat}}} C \tag{5.3}$$

where C is the fixed cycle time and r_{sat} is the ramps saturation flow. The green-phase duration is constrained by $g(t) \in [g_{\min}, g_{\max}]$, where $g_{\min} > 0$ to avoid ramp closure, and $g_{\max} < C$. In the case of an one-car-per-green realization (quite popular in USA), a constant-duration green phase permits exactly one vehicle to pass. Thus, the ramp volume is controlled by varying the red-phase duration between a minimum and a maximum value. Note that ALINEA is also applicable directly to the green or red-phase duration, by combining equations (5.2) and (5.3)

$$g(t) = g(t-1) + K'_{\text{R}} \left[\widehat{o} - o_{\text{out}}(t)\right], \quad \text{with} \quad K'_{\text{R}} = K_{\text{R}} \frac{C}{r_{\text{sat}}} \tag{5.4}$$

Note also, that the values $r(t-1)$ or $g(t-1)$ used on the right-hand side of equation (5.2) or (5.4), respectively, should be the **bounded** values of the previous time step (i.e., after application of the $g_{\min}$ and $g_{\max}$ constraints), in order to avoid the windup phenomenon in the I-regulator. Finally, if the queue of vehicles on the ramp becomes excessive, interference with urban street traffic may occur. This may be detected with suitably placed detectors (on the upstream part of the on-ramp), leading to an override of the regulators decisions to allow more vehicles to enter the motorway and the ramp queue to diminish.

Comparative field trials have been conducted in various countries to assess and compare the efficiency of local ramp metering strategies. It has been seen that ALINEA outperforms the feedforward-based demand-capacity and occupancy strategies, with respect to all evaluation criteria (see [60] for details on field applications results).

5.2.3 Multivariable regulator strategies

Multivariable regulators for ramp metering pursue the same goals as local ramp metering strategies: they attempt to operate the motorway traffic conditions near some prespecified set (desired) values. While local ramp metering is performed independently for each ramp, based on local measurements, multivariable regulators make use of all available mainstream measurements $o_i(t), i = 1, \ldots, n$, on a motorway stretch, to calculate simultaneously the ramp volume values $r_i(t), i = 1, \ldots, m$, for all controllable ramps included in the same stretch [57].

This provides potential improvements over local ramp metering because of more comprehensive information provision and coordinated control actions. Multivariable regulator approaches to ramp metering have been mostly derived by application of the Linear-Quadratic-Regulator (LQR) theory ([47], [25], [56], [76]). The multivariable regulator strategy METALINE may be viewed as a generalization and extension of ALINEA, whereby the metered on-ramp volumes are calculated from

$$r(t) = r(t-1) + K_1 \left[o(t) - o(t-1)\right] + K_2 \left[\widehat{O} - O(t)\right] \tag{5.5}$$

where $r = [r_1, r_2, \ldots, r_m]^T$ denotes the vector of the m controllable on-ramp volumes, $o = [o_1, o_2, \ldots, o_n]^T$ denotes the vector of the n measured occupancies on the motorway stretch, $O = [O_1, O_2, \ldots, O_m]^T$ is a subset of o that includes m occupancy locations for which prespecified set values $\widehat{O} = \left[\widehat{O}_1, \widehat{O}_2, \ldots, \widehat{O}_m\right]^T$ may be given. Note that for control-theoretic reasons, the number of set-valued occupancies cannot be higher than the number of controlled on-ramps. Typically, one bottleneck location downstream of each controlled on-ramp is selected for inclusion in the vector O. Finally, K_1 and K_2 are the regulators constant gain matrices that must be suitably designed (see [57] for details).

Field trials and simulation results comparing the efficiency of METALINE versus ALINEA have led to the following conclusions:

- While ALINEA requires hardly any design effort, METALINE application calls for a rather sophisticated design procedure that is based on advanced control-theoretic methods (LQR optimal control).

- For urban motorways with a high density of on-ramps, METALINE was found to provide no advantages over ALINEA (the latter implemented independently at each controllable on-ramp) under recurrent congestion.

- In the case of nonrecurrent congestion (e.g. due to an incident), METALINE performs better than ALINEA due to more comprehensive measurement information.

Some system operators hesitate to apply ramp metering because of the concern that congestion may be conveyed from the motorway to the adjacent urban road network. In fact, a ramp metering application designed to avoid or reduce congestion on motorways may have both positive and negative effects on the adjacent road network traffic. It is easy to see, based on notions and statements made earlier, that, if an efficient control strategy is applied for ramp metering the motorway throughput will be generally increased.

More precisely, ramp metering at the beginning of the rush hour may lead to on-ramp queues in order to prevent congestion to form on the motorway, which may temporarily lead to diversion toward the urban network. Nevertheless, due to congestion avoidance or reduction, the motorway will be eventually enabled to accommodate a higher throughput, thus attracting drivers from urban paths and leading to an improved overall network performance. This positive impact of ramp metering, on both the motorway and the adjacent road network traffic conditions, was confirmed in a specially designed field evaluation in the Corridor Périphérique in Paris (see [26]).

5.3 The macroscopic simulator METANET

Modeling of traffic flow on motorway networks is a useful tool for several traffic engineering tasks such as:

- Development and evaluation of traffic control strategies.
- Short-term prediction and surveillance of traffic state in complex networks.
- Evaluation of the impact of new constructions, comparison of alternatives, etc.
- Evaluation of the impact of capacity reducing events (e.g. roadworks) or increased demand, etc.

According to the specific task, the use of the model may be either off-line or in real-time.

Macroscopic modeling of motorway traffic flow implies the definition of adequate variables expressing the aggregate behavior of traffic at certain times and locations. For the macroscopic description of traffic flow on a motorway link or a motorway network, the classical mathematical tools of state differential equations (e.g. $\dot{x} = f\,[x, u, d]$) or difference equations (e.g. $x(t+1) = f\,[x(t), u(t), d(t)]$) are perfectly suitable.

METANET is a software for motorway network simulation based on a purely macroscopic modeling approach. A validated macroscopic second-order traffic flow model ([53]) is used for the simulation of traffic flow. The time and space arguments are discretized. A motorway link is divided into segments of equal length (typically 500m). The traffic in each segment of the

link at every discrete-time index is macroscopically characterized via the following variables: the traffic density (veh/km/lane), the traffic volume or flow (veh/h) and the mean speed (km/h). This leads to relatively low computational effort, which is independent of the load (number of vehicles) in the simulated network and allows also for a real-time use of the model. The overall modeling approach, allows for simulation of all kinds of traffic conditions (free, dense, congested) and of capacity reducing events (incidents) with prescribed characteristics (location, intensity, duration).

METANET, may be applied to existing or partially hypothetical, multi-origin, multi-destination, multi-route motorway networks with arbitrary topology and geometric characteristics including bifurcations, junctions, on-ramps and off-ramps. The motorway network is represented by a directed graph, whereby the links of the graph represent motorway stretches. Each stretch has uniform characteristics, i.e. no on-/off-ramps and no major changes in geometry. The nodes of the graph are placed at locations where a major change in road geometry occurs, as well as at junctions, on-ramps and off-ramps. By use of a special modeling option (store-and-forward links [23]), METANET provides also the possibility to consider non-motorway links in a simplified way.

METANET considers the application of traffic control measures, such as collective and/or individual route guidance as well as ramp metering and motorway-to-motorway control, at arbitrary network locations. Several options are offered for describing or prescribing the average route choice behavior of drivers groups with particular destinations.

Simulation results are provided in terms of macroscopic traffic variables such as traffic density, traffic volume, and mean speed at all network locations on a chosen time interval basis (typically 5 to 20s; although the output interval may be chosen longer). Visualization of results is provided by time trajectories of selected variables and by graphical representation of the whole network. Global evaluation indexes such as total travel time, total traveled distance, total fuel consumption, total waiting time at network origins, etc. are also calculated.

Finally, for displaying traffic data generated or used by METANET in a transparent form, an extra graphical output program called METAGRAF is available.

5.4 The Monash motorway in Melbourne

The first test case for the simulation experiments is a part of the Monash–CityLink–West Gate Corridor in Melbourne, Australia, which is operated by VicRoads. Figure 5.4 shows a map of the motorway stretch extending from Jacksons RD to Warrigal RD. The considered direction is the westbound which leads to the city centre. The total length of the modeled motorway stretch is about 17km and contains a total of 8 on-ramps and 7 off-ramps. Congestion usually

appears right downstream Burke RD and spills back creating severe shockwaves.

The total number of links used to model the motorway is 34. This number includes 17 motorway links, 9 origin links and 8 destination links. The motorway links have been divided into a total of 38 segments with lengths that range from 300m to 650m. At the present time, an ALINEA/HERO [61] ramp metering control strategy is implemented at this motorway stretch.

Figure 5.5 depicts the resulting simulation model of the motorway stretch along with the positions of the detector stations (bullet points with associated ids). Arrows represent links, and circles represent nodes. Three types of links are used, namely, motorway links (L1-L17), origin links (on-ramp names and MONASH_FR_ORIGIN) and destination links (off-ramp names and MONASH_FR_DESTIN). Each motorway link has been divided into a number of segments bounded in Figure 5.5 with vertical lines.

A node is used mainly whenever there is an origin or a destination that needs to be connected with the mainstream. Node N13 has been used in order to break the corresponding long stretch into two links. This is necessary as the main bottleneck of this motorway stretch is located in the area of L13a, where very intensive lane changes take place and the capacity appears to be lower than the one on L13. The used model does not explicitly take into account lane changes. As a result, the only way to enable the creation of congestion in this area is the use of a different fundamental diagram, i.e. a different capacity value. All links up to L13a have 3 lanes and all links from L14 and downstream have 4 lanes.

5.5 Application of AFT to ALINEA strategy

In order to evaluate the efficiency of the AFT algorithm, presented in Chapter 4, to the problem of optimizing the design parameters of a ramp metering system applied to the Monash motorway, extensive simulation experiments have been conducted using the macroscopic traffic simulation tool METANET. The traffic model parameters assumed in METANET were obtained by use of nonlinear parameter estimation techniques [61], minimizing the mismatch between the METANET states and actual traffic data provided by VicRoads.

It has to be emphasized that the problem of computing the "optimal" design parameters for this particular control system can be formulated as a nonlinear optimization problem, the solution of which depends on the traffic demand (vehicles entering the motorway network). As a result, even in the case where the system dynamics are exactly known the computation of the "optimal" control design parameters is a NP-hard problem, and, moreover, it requires knowledge of the traffic demand. It is also noted that – after a tedious and time-consuming fine-tuning – the implementation of the investigated control system in various motorway

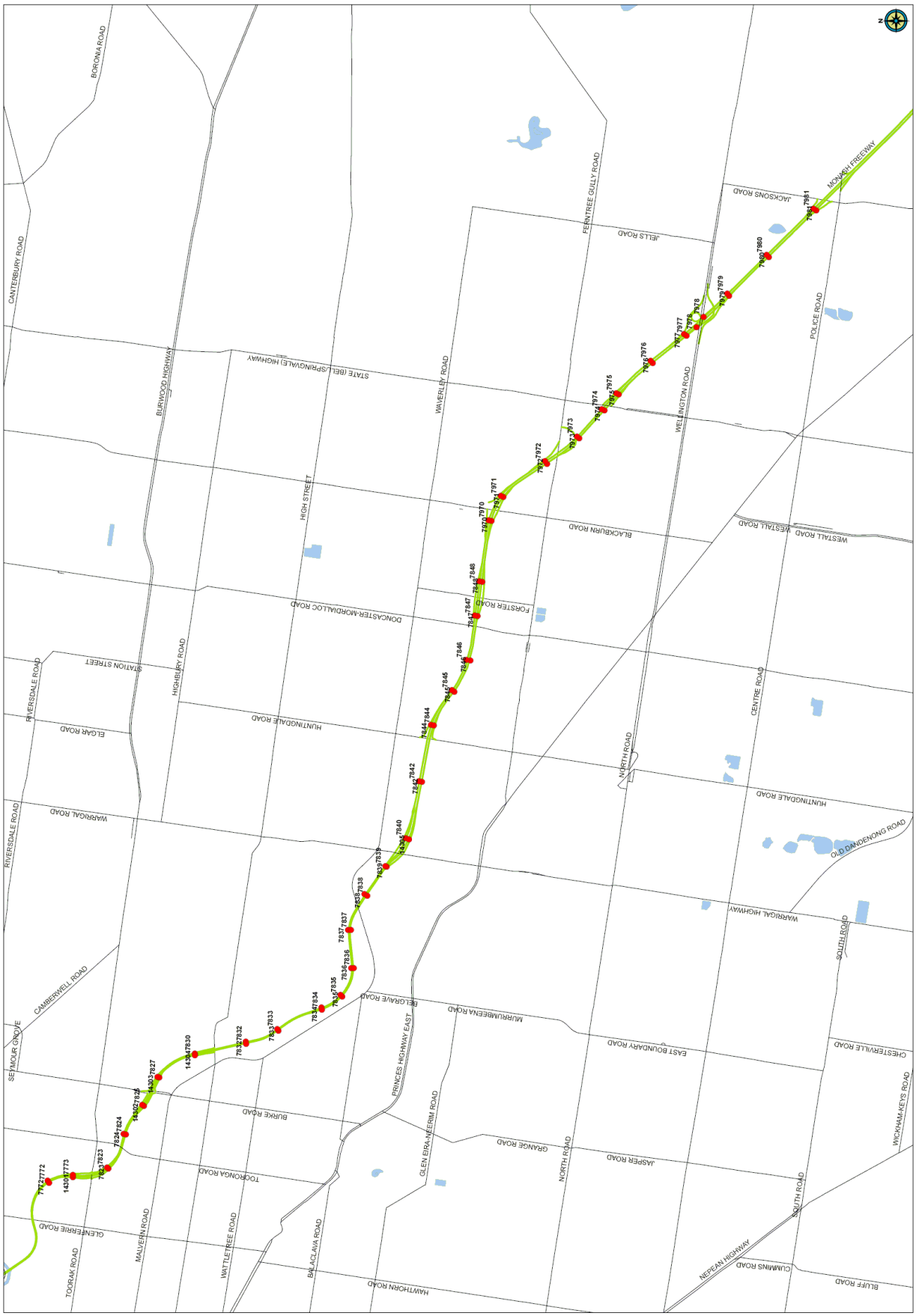

Figure 5.4: The Monash motorway stretch in Melbourne. The considered direction is the westbound.

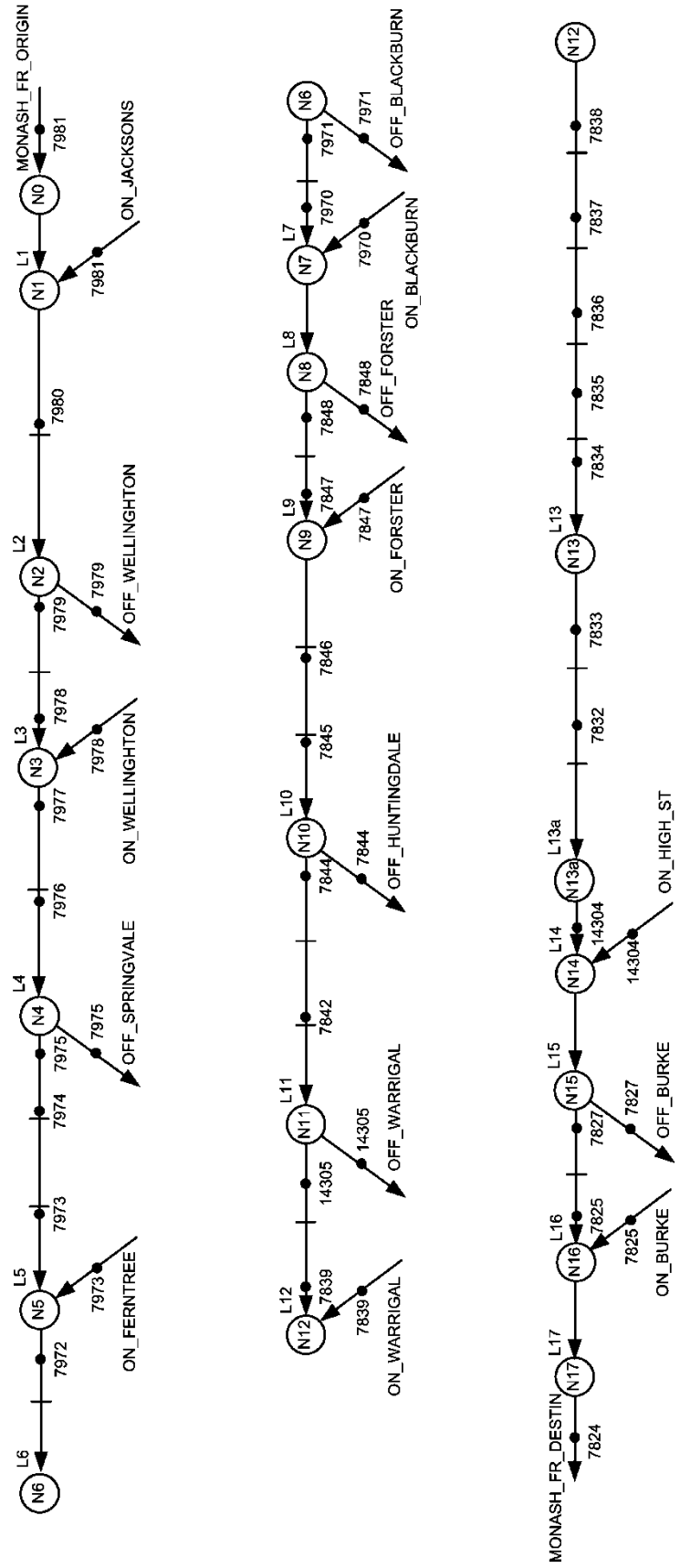

Figure 5.5: Representation of the Monash motorway stretch simulation model.

networks produced very efficient performance (see [58] for details).

In the simulation experiments controlled ramp metering is imposed at all ramps except ramp ON_BURKE (Figure 5.5), since congestion appears upstream that ramp and spills back due to waving at the end of link 13a (because of the lane changing), creating severe shockwaves. The purpose of the applied ramp metering control is to take into account spill-back of congestion from the area of Burke RD and study the effect of the developed coordination scheme on the traffic conditions in this area. In the next two sections, the details regarding the particular LNTCS application are presented.

5.5.1 Tunable parameters

The form of the motorway ramp metering regulator applied in the simulation experiments reads

$$r(t) = H\left(r(t-1) - K_1 o(t) + K_2\right), r(0) = r_{\max}. \tag{5.6}$$

The control input vector $r(t)$ denotes the ramp flows allowed through the implementation of ramp metering at the discrete time-index t and $o(t)$ denotes the vector of average measured occupancies at the detector locations downstream the controlled ramps. Vector $r_{\max}$ denotes the maximum admissible flow for all the ramps. The occupancy measurements of detectors 7980, 7977, 7972, 7848, 7846, 7838 and 7827 are used to form the vector $o(t)$. The nonlinear (saturation) operator H is used to guarantee that the control decisions satisfy minimum and maximum admissible ramp flow constraints.

Equation (5.6) is a multivariable feedback regulator and may be considered as a generalization of the local strategy ALINEA (equation (5.2) presented in Section 5.2.2). It is worth noting, that (5.6) has exactly the form of (5.2) with the matrix K_1 being a diagonal matrix and the vector K_2 depending on the critical densities of the locations downstream the on-ramps. Also, the strategy METALINE (equation (5.5) in section 5.2.3) and a variety of other multivariable (network-wide) ramp metering strategies have the form of (5.6), in which cases matrices K_1, K_2 are calculated using nonlinear optimization or optimal control techniques.

The entries of K_1 and K_2 affect the performance of (5.6), thus were selected for automated fine-tuning by the AFT algorithm. According to the notations of Chapter 4, for this application the vector of tunable parameters θ has dimension $7 \times 7 + 7 = 56$ and is equal to $\theta = \mathrm{vec}\,(\theta_1, \theta_2)$, where θ_1 a vector with the 49 entries of matrix K_1 and θ_2 a vector with the 7 entries of matrix K_2.

According to the initial values of these parameters, two different control scenarios are considered which are both described in details in the following section. For all the ramp

metering experiments the cycle time period t is set equal to 30sec. In order to assess the overall system performance the criterion $J \equiv ms$ (equation (4.3)) was set to the actual daily network mean speed. Finally, a projection to the nearest valid point is applied whenever a vector θ violates the imposed constraints.

5.5.2 Demand scenario and estimation of exogenous signals

Based on actual traffic data measurements provided by VicRoads, a **basic daily traffic demand scenario** $D_z(t_d)$ was designed (where $D_z(t_d)$ denotes the number of vehicles entering the z-th motorway origin at the t_d-th interval), with duration equal to 8 hours (only the "peak hours period", i.e. the period of high traffic demand within the day was considered in the simulations). At each fine-tuning experiment (considered as one simulated day), a random perturbation of the basic scenario was used. More precisely, at each fine-tuning experiment, the traffic demand was calculated according to

$$D_z^{(k)}(t_d) = \max\{0, D_z(t_d) + D_z(t_d)w_z^{(k)}(t_d)\}, \tag{5.7}$$

where $D_z^{(k)}(t_d)$ denotes the traffic demand at the k-th fine-tuning experiment and $w_z^{(k)}(t_d)$ is a Gaussian zero-mean term with variance equal to 0.1. It is worth noting that the basic demand scenario corresponded to highly congested traffic conditions. Note also, that due to the use of the Gaussian random term $w_z^{(k)}(t_d)$, the exogenous signal $x(k)$ – whose entries correspond to the elements of $D_z^{(k)}(t_d)$ – is an **unbounded** signal in the sense that it is not possible to a priori assess an upper bound for its magnitude.

The average mean speed of the whole traffic network (in km/h) over the 8 hours was used as the performance metric to be optimized by the fine-tuning algorithm. It is worth noting that the average mean speed can be calculated based on detector measurements. Note also, that since the goal of a traffic control system is to maximize mean speed, performance index maximization (by appropriately modifying the proposed algorithm) instead of minimization was implemented.

The exogenous vector $x(k)$ in this particular application corresponds to the traffic demand $D_z^{(k)}(t_d)$ (time-history of the number of vehicles entering the network in each of the network origins). Given the fact that there are 9 origins in the network and a daily demand scenario corresponds to 8 hours, it would be computationally cumbersome to use vectors with entries that correspond to estimates at small time-intervals (e.g., if the entries of $x(k)$ correspond to 3-min estimates, the total size of these vectors is equal to 1440 entries). On the other hand, given that the demand estimate for the next day will be anyway inaccurate (due to a natural variation of the demand), it does not make sense to use estimates over small time-intervals.

Moreover, the dimension n_x of the estimate vector $\bar{x}(k)$ should be comparative to the dimension n_θ of the vector of tunable parameters θ. In our experiments, we found that it suffices to use an estimate every 1h (i.e., eight different estimates for the demand of the whole day). This low-dimension, noisy estimate $\bar{x}(k)$ of the traffic demand was constructed based on the average number of vehicles entering every particular origin over the 1-h period and resulted in vectors of dimension $n_x = 72$.

5.6 Simulation results

In order to evaluate the efficiency of the presented AFT algorithm to the problem of optimizing the design parameters of an ALINEA-based ramp metering system, extensive simulation experiments have been conducted. The performance of AFT is compared to the base-case (no AFT case) for two different control scenarios: (1) when the initial values of the tunable parameters $\theta(0)$ are all set equal to 0 (assuming that there is no information available about the system or the applied regulator), and (2) when the initial values of the tunable parameters $\theta(0)$ are selected according to some empirical rules or any other available information.

5.6.1 Global results for control scenario 1

This section presents the simulation results obtained for the control scenario 1 described above. The initial values for the vector θ are all set to 0. As a result, the no AFT case (derived from equation (5.6) for $K_1, K_2 = 0$) gives $r(t) = r(0) = r_{\max}, \forall t$, which is similar to the no control case, as the ramp flows do not change at all. The values of vector $r_{\max}$ are set big enough, so as to correspond to maximum green times for all the controlled ramps. Alternatively, AFT changes the values of θ iteration-by-iteration trying to locate a good set of parameters (according to the simulated daily mean speed of the network).

Because of the daily stochastic fluctuation of the demand scenario, 10 different simulation runs were carried out for statistical purposes. Figures 5.6–5.10 compare the network-wide mean speed of the no AFT case (blue line) versus 2 runs of the AFT algorithm (red and green lines) delivered for control scenario 1. The only difference between two successive runs of AFT algorithm is the stochasticity of the demand (randomly perturbed $\pm 10\%$ at every iteration). In all figures, it can be seen that the application of the AFT algorithm leads to better performance than the no control case. More precisely, the AFT algorithm achieves to persistently keep the mean speed of the network higher than the no AFT case during the whole fine-tuning period.

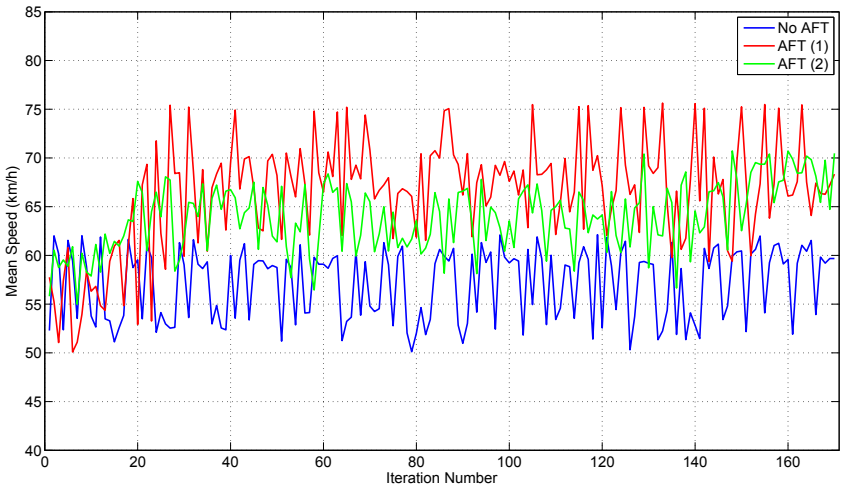

Figure 5.6: Mean speed for control scenario 1 (no control/AFT–run 1 and 2).

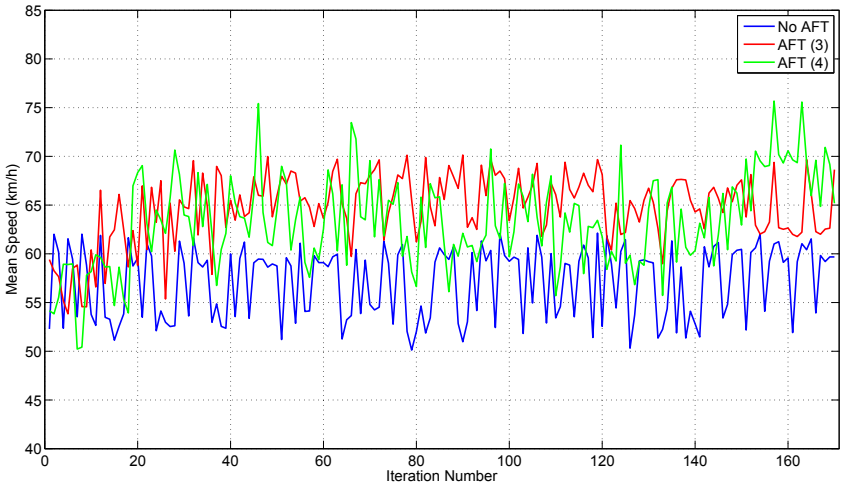

Figure 5.7: Mean speed for control scenario 1 (no control/AFT–run 3 and 4).

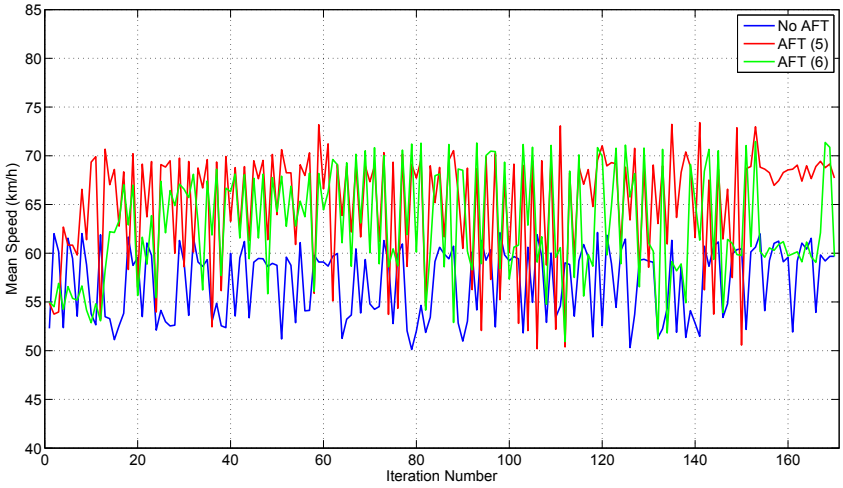

Figure 5.8: Mean speed for control scenario 1 (no control/AFT–run 5 and 6).

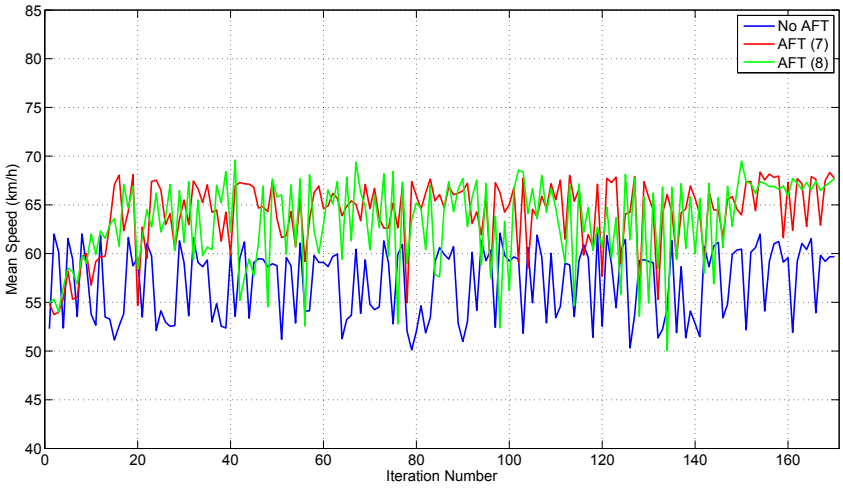

Figure 5.9: Mean speed for control scenario 1 (no control/AFT–run 7 and 8).

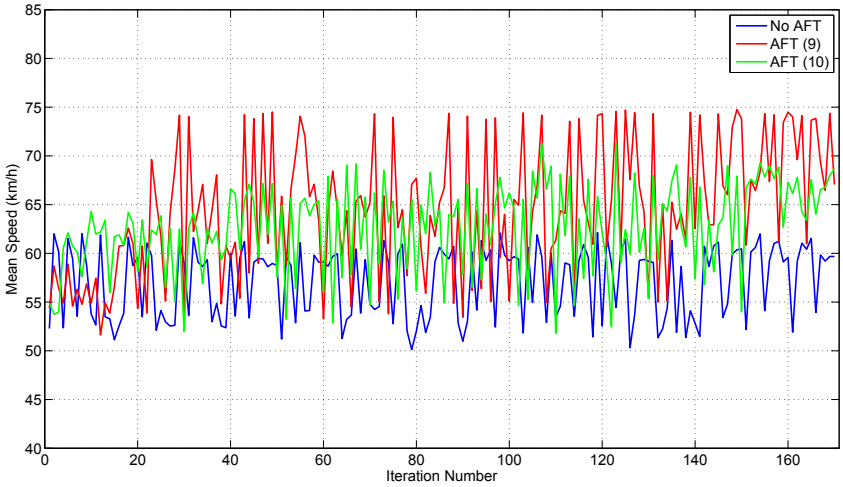

Figure 5.10: Mean speed for control scenario 1 (no control/AFT–run 9 and 10).

All these simulation experiments were conducted with a constant stepsize $\alpha(k) = \alpha = 0.1$. Thus, AFT does't converge to a "good" set of parameters, albeit, it continues searching throughout the whole simulation horizon. Accordingly, the trajectories of the system mean speed (red and green lines in Figures 5.6–5.10) keep fluctuating, sometimes with a higher rate than the no AFT case (blue line). The blue line fluctuates only because of the demand changes, the red and green lines, however, because of both the demand changes and the search process of AFT algorithm.

Table 5.1 displays the mean speed for the no AFT case and the mean speed when using AFT for the system fine-tuning, for each simulation run. Also, the standard deviation of the mean speed due to the $\pm 10\%$ daily random perturbations of the demand scenario is

Table 5.1: Comparison of the average mean speed (MS) with and without the application of AFT algorithm to ramp metering (control scenario 1).

	All 170 days		Conv.	After convergence day		
AFT	MS (km/h)	Improvement	Day	MS (km/h)	St.Dev.	Improvement
1	66.38	16.26%	22	67.67	4.35	18.51%
2	63.98	12.05%	18	64.46	3.26	12.89%
3	64.69	13.29%	12	65.16	2.99	14.12%
4	63.34	10.92%	20	64.15	4.23	12.35%
5	65.09	13.99%	8	65.36	5.79	14.46%
6	62.86	10.08%	17	63.51	5.30	11.22%
7	64.08	12.22%	14	64.63	2.95	13.18%
8	63.24	10.76%	14	63.63	4.18	11.44%
9	64.37	12.73%	23	65.41	6.51	14.56%
10	62.68	9.78%	10	62.92	4.52	10.19%
Average	64.07	12.21%	15.8	64.69	4.41	13.29%

presented. It should be noted, that AFT algorithm is learning the traffic system dynamics during the first iterations (days) by experimenting with different sets of parameters. This means, that the results of the first few days of implementation (see Figures 5.6–5.10) do not derive from a well-defined optimization procedure, because of the lack of data to be used by the approximator. After the collection of several input-output data sets (after the simulation of some days) AFT can use the data to converge to a local optimum. Thus, for an assessment of the algorithm's convergence we define

$$\text{ConvDay} = \arg\min_{d} \mathcal{J}\left(J(d) \geq 0.9J^*\right) \tag{5.8}$$

where ConvDay is the day that we assume that the algorithm has converged, J^* denotes the maximum achievable performance (mean speed) throughout the whole simulation horizon and $\mathcal{J}$ denotes a set with the outputs of all simulations.

The ConvDay is also shown in Table 5.1 for all simulation runs (15.8 days on average) and the comparison of AFT to the no AFT case is presented for both the whole 170 simulated days and if the learning period is excluded. The use of AFT algorithm leads to an average improvement of the system performance of some 12% for this scenario, and if the learning period is excluded the improvement is some 14%.

For the same control scenario the SPSA algorithm was applied in order to fine-tune the

systems's design parameters. The form of the applied iterative algorithm reads

$$\theta(k+1) = \begin{cases} \theta^*(k) + \alpha(k)\Delta(k), & \text{if } k \text{ odd} \\ \theta(k) + \beta(k)\widehat{\nabla J(k)}, & \text{otherwise} \end{cases} \tag{5.9}$$

with

$$\widehat{\nabla J(k)} = \frac{J(k) - J(k-1)}{\Delta\theta(k)} \tag{5.10}$$

where $\Delta\theta(k) = \theta(k) - \theta(k-1)$ and $\theta^*(k)$ the best set of tunable parameters found so far. The above iterative method applies a random perturbation of the regulator's parameter vector every odd day and uses the last two days measurements to obtain an approximation of the performance index gradient (applying a gradient-ascent modification based on the gradient estimate).

Table 5.2 displays the obtained results for 6 different simulation runs of SPSA. Different constant values where applied for the gain sequences $\alpha(k), \beta(k)$. It is clear that SPSA cannot increase the mean speed of the network. The scale and complexity of the fine-tuning problem prevents equation (5.10) from providing a good estimation of the system's performance gradient. Thus, the applied sets of parameters by SPSA algorithm can be characterized as "random", as they fail to achieve any improvement to the system performance. The average mean speed for the no AFT case is 57.10 km/h and for SPSA 57.42 km/h. Figures 5.11 and 5.12 present the trajectories of the mean speed for all 6 runs of SPSA algorithm.

Table 5.2: Comparison of the average mean speed (MS) with and without the application of SPSA algorithm to ramp metering (control scenario 1).

		SPSA						
	No AFT	1	2	3	4	5	6	Average
MS (km/h)	57.10	58.02	57.61	57.36	57.07	57.10	57.36	57.42
St.Dev.	2.01	1.69	1.89	1.98	1.94	2.01	1.98	1.91
Improvement		1.61%	0.90%	0.46%	-0.06%	0.00%	0.46%	0.56%

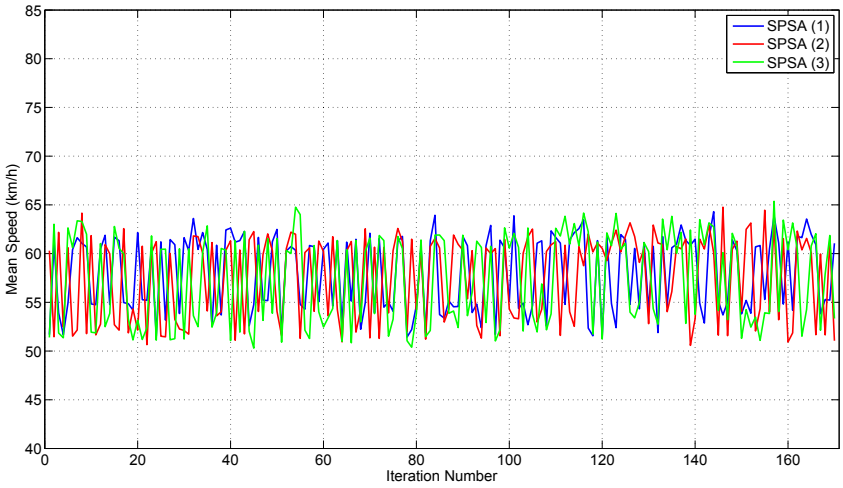

Figure 5.11: Mean speed for control scenario 1 (no control/SPSA–runs 1, 2 and 3).

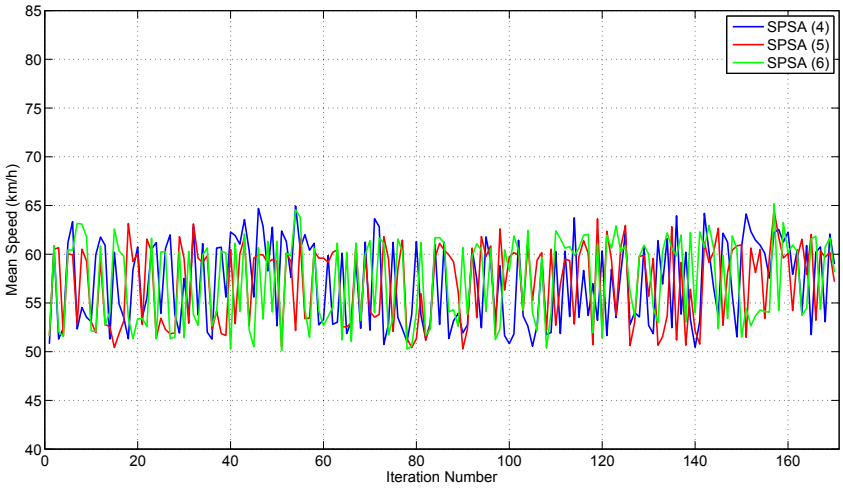

Figure 5.12: Mean speed for control scenario 1 (no control/SPSA–run 4, 5 and 6).

5.6.2 Global results for control scenario 2

This section presents the simulation results obtained for the control scenario 2. The initial values for the vector θ are selected according to $K_1 = I_7$ and $K_1 = [25, 25, 25, 25, 25, 25, 25]^T$, with I_7 denoting the 7x7 unit matrix (recall that $\theta = \text{vec}\,(\theta_1, \theta_2)$, where θ_1 a vector with the 49 entries of matrix K_1 and θ_2 a vector with the 7 entries of matrix K_2). As a result, the no AFT case (see equation (5.6)) applies 7 local ALINEA strategies to the controlled ramps (with the critical occupancy equal to 0.25 for all ramps). Alternatively, AFT changes the values of θ iteration-by-iteration trying to determine a good multivariable regulator (according to the feedback of the network-wide simulated daily mean speed).

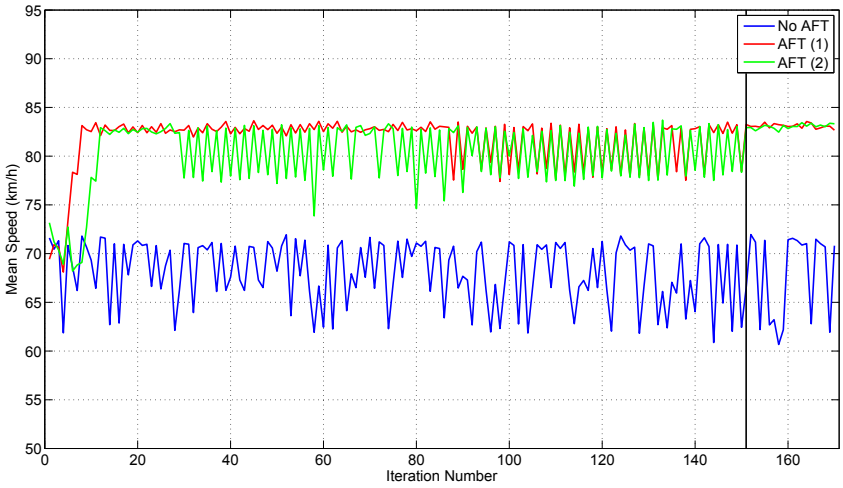

Figure 5.13: Mean speed for control scenario 2 (ALINEA/AFT–run 1 and 2).

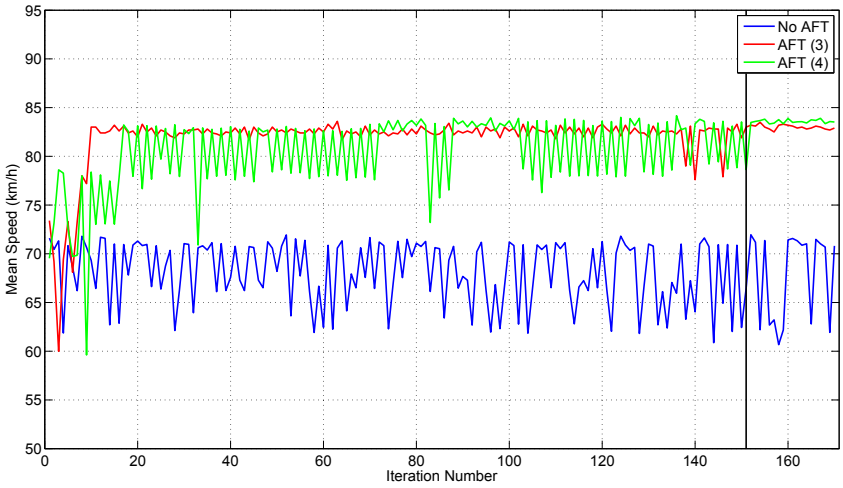

Figure 5.14: Mean speed for control scenario 2 (ALINEA/AFT–run 3 and 4).

Because of the daily stochastic fluctuation of the demand scenario, 10 different simulation runs were carried out for statistical purposes. Figures 5.13–5.17 compare the network-wide mean speed of the no AFT case (blue line) versus 2 runs of AFT algorithm (red and green lines) delivered for this control scenario. Again, the only difference between two successive runs of AFT is the stochasticity of the demand (randomly perturbed $\pm 10\%$ at every iteration). In all figures, it can be seen that the application of AFT algorithm leads to better performance than the application of 7 local ALINEA strategies. More precisely, the AFT algorithm achieves to optimize the overall system performance within few days (iteration number in figures), by efficiently fine-tuning the regulator's design parameters, while avoiding decreasing the daily mean speed lower than the no AFT case. The trajectories of the mean speed are persistently increasing in all figures, until they converge to a local maximum value.

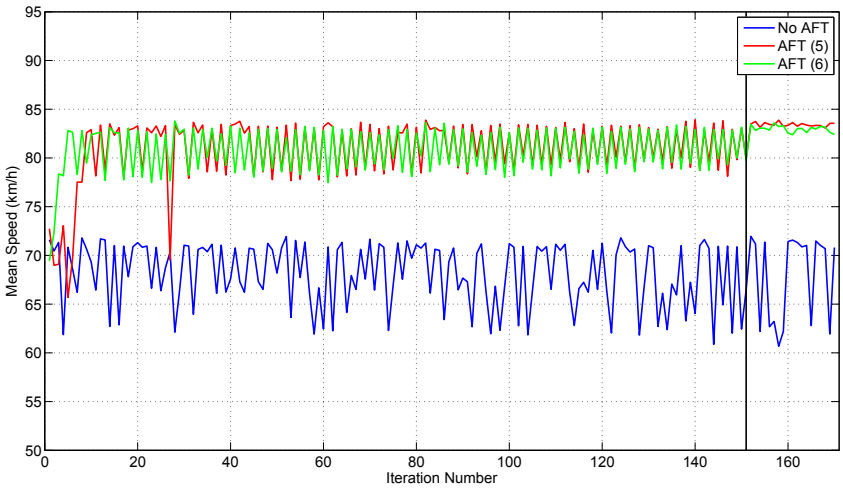

Figure 5.15: Mean speed for control scenario 2 (ALINEA/AFT–run 5 and 6).

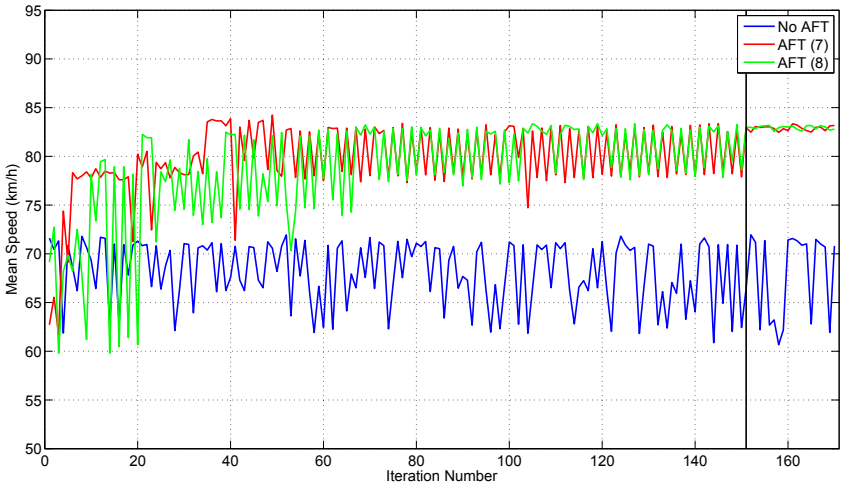

Figure 5.16: Mean speed for control scenario 2 (ALINEA/AFT–run 7 and 8).

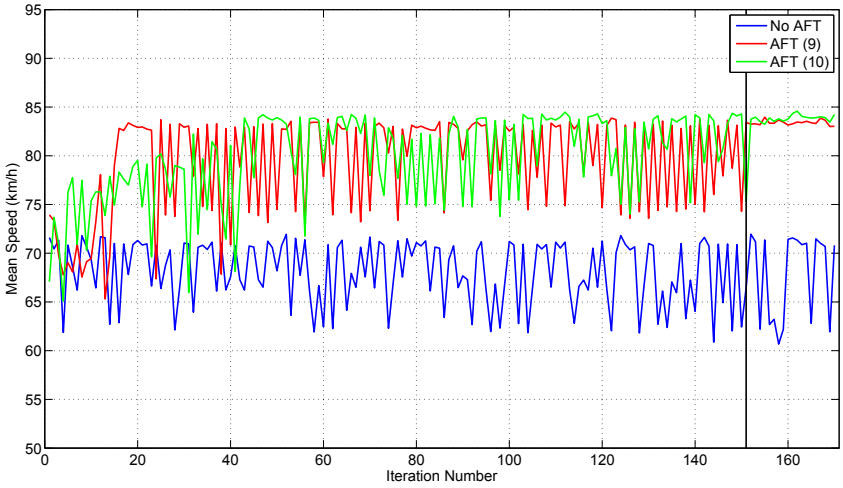

Figure 5.17: Mean speed for control scenario 2 (ALINEA/AFT–run 9 and 10).

In all simulation runs, equations (4.23)–(4.24) were applied for the stepsize of AFT algorithm, with $\alpha(0) = 1$. The descending sequence of $\alpha(k)$ leads to a smoother algorithm convergence compering to the case of constant stepsize (scenario 1). The proposed formula has the advantage that after some iterations descends to small values for the stepsizes $\alpha(k)$, providing the algorithm with a solution close to the optimal solution found so far. Again, there are some fluctuations due to the random demand changing.

Table 5.3 displays the mean speeds and the standard deviations for control scenario 2. It can be seen that the use of AFT algorithm leads to an average improvement of some 18% for this demand scenario, which is increased to some 19% if the learning period is excluded (7 days on average). The variations between different runs are low. The average mean speed for the original TUC system is 68.11 km/h and after the convergence of AFT this speed is increased to 81.11 km/h. Finally, it is clear that the descending stepsize sequence helps the convergence of the algorithm.

Table 5.3: Comparison of the average mean speed (MS) with and without the application of AFT algorithm to ALINEA strategy (control scenario 2).

	All 170 days		Conv.	After convergence day		
AFT	MS (km/h)	Improvement	Day	MS (km/h)	St.Dev.	Improvement
1	81.80	20.10%	6	82.13	1.91	20.59%
2	80.49	18.18%	10	81.04	2.60	18.98%
3	81.96	20.33%	8	82.55	1.15	21.20%
4	80.77	18.59%	3	81.48	3.67	19.63%
5	81.13	19.11%	7	81.54	2.53	19.72%
6	81.03	18.98%	3	81.16	2.20	19.17%
7	80.10	17.61%	6	80.20	2.85	17.76%
8	79.16	16.22%	10	79.85	4.65	17.24%
9	79.76	17.11%	12	80.52	4.31	18.23%
10	80.28	17.87%	5	80.67	4.24	18.44%
Average	80.65	18.41%	7	81.11	3.01	19.09%

For the same control scenario the SPSA algorithm was applied (as described in equations (5.9)–(5.10)) in order to fine-tune the systems's design parameters. Table 5.4 displays the obtained results for 6 simulation runs with different values of the parameters $\alpha(k)$ and $\beta(k)$. It is clear that SPSA cannot increase the mean speed of the network. The scale and complexity of the fine-tuning problem doesn't allow for a good estimation of the gradient, considering only one simultaneous perturbation. Thus, the resulting sets of parameters provided by SPSA algorithm fail to achieve any improvement to the system performance. The average mean speed without SPSA is 68.11 km/h and with SPSA 69.42 km/h. Finally, Figures 5.18 and 5.19 present the trajectory of the mean speed for the aforementioned simulation runs.

Table 5.4: Comparison of the average mean speed (MS) with and without the application of SPSA algorithm to ALINEA strategy (control scenario 2).

		SPSA						
	No AFT	1	2	3	4	5	6	Average
MS (km/h)	68.11	69.86	69.06	70.61	70.05	68.08	68.87	69.42
St.Dev.	4.84	2.66	4.58	3.47	4.14	5.98	4.36	4.20
Improvement		2.57%	1.40%	3.67%	2.85%	-0.04%	1.11%	1.93%

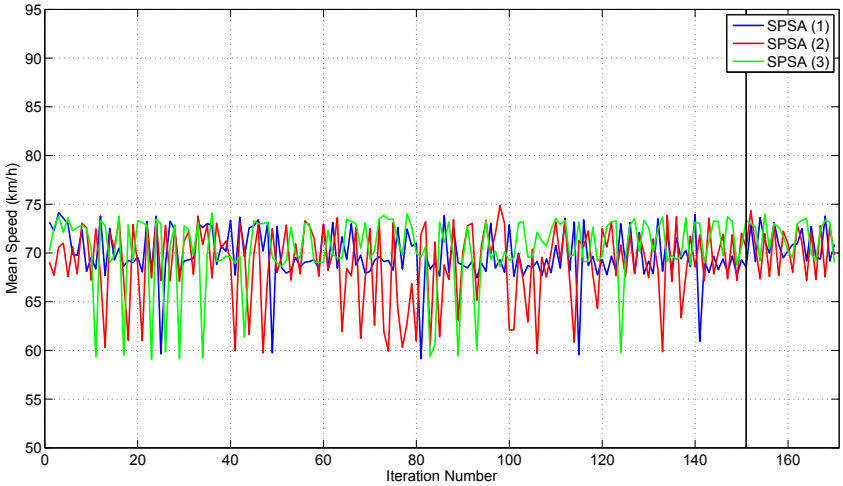

Figure 5.18: Mean speed for control scenario 2 (ALINEA/SPSA–run 1, 2 and 3).

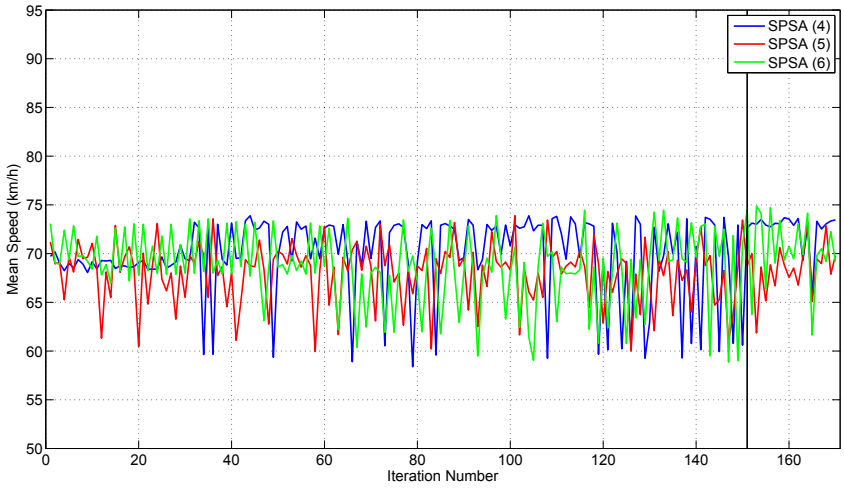

Figure 5.19: Mean speed for control scenario 2 (ALINEA/SPSA–run 4, 5 and 6).

5.7 Concluding remarks

This chapter investigated the efficiency of AFT algorithm when applied to the problem of optimizing the design parameters of a large-scale ramp metering control system. This adaptive optimization methodology aims at replacing the conventional manually-based optimization with a fully-automated procedure. Extensive simulation experiments have been conducted for the ramp metering control problem of the large-scale traffic network of Monash motorway in Australia, where the design parameters of an ALINEA-based multivariable approach were fine-tuned by AFT algorithm. The simulation results, as well as the comparison to the base-case, where the aforementioned design parameters were selected manually, demonstrate the algorithm's efficiency and feasibility.

Two different control scenarios have been considered for fine-tuning. In the first one, it was assumed that we have no knowledge about the controlled system at all. All the initial values of the regulator's design parameters were set equal to 0, letting AFT algorithm to provide an efficient multivariable regulator. A comparison to the no control case was used in order to evaluate the results. Regarding the second scenario, we investigated the improvement of applying AFT algorithm to 7 local ALINEA regulators, in order to come up with a network-wide multivariable regulator after the fine-tuning procedure (by appropriately adjusting the gain matrices of equation (5.6)).

Figure 5.20 illustrates a conclusive remark about this application, based on the average obtained results. For the selected demand scenario (which is quite realistic, since it is based on real measurements) the no control case corresponds to an average network mean speed of 57.10 km/h. When AFT is applied for fine-tuning to a (zero knowledge) multivariable regulator the mean speed is increased to 64.69 km/h (13.29%), whereas SPSA cannot actually

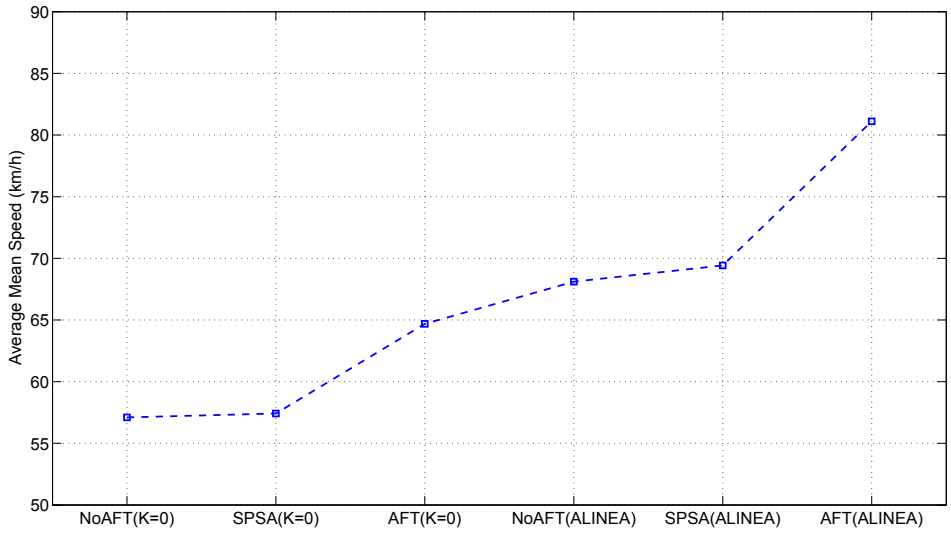

Figure 5.20: Conclusive (average) results for the application of AFT algorithm to ramp metering control.

provide any improvement when applied to the same regulator (just 0.56%, with average network mean speed 57.42 km/h). When 7 local ALINEA regulators are applied to the 7 controlled ramps, the average mean speed of the network is increased to 68.11 km/h. Note that this speed is 5.29% higher than the one after the convergence of AFT in the first control scenario. That is, AFT converges to a local optimal solution which provides a multivariable regulator that is worse than the 7 local ALINEA regulators. Finally, the fine-tuning of the regulator of the second control scenario leads to an additional improvement of 19.09% (network mean speed 81.11 km/h) after the convergence of the algorithm. Again SPSA fails to produce a notable improvement, resulting in a mean speed of 69.42 km/h on average (1.93% improvement).

It is worth noting, that the average computational time for every iteration of AFT algorithm is a few seconds, which means that the implementation of the algorithm in an online real-time large-scale application would be feasible, regardless the type of the operating Traffic Control System.

Chapter 6

Application to urban traffic control

This chapter presents the application results of AFT algorithm to a large-scale urban signal control problem. Section 6.1 introduces the reader to the concept of traffic management and Urban Traffic Control (UTC) systems. Section 6.2 provides a short review of the most popular UTC systems worldwide, while Section 6.3 focuses on the UTC strategy TUC and its control modules. In Section 6.4 the microscopic simulator AIMSUN used for the simulation experiments is presented, while in Section 6.5 a short description of the studied urban road network of the city of Chania, in Greece is provided. Following, Section 6.6 discusses the application set-up of the algorithm for this particular network implementation and Section 6.7 thoroughly examines and analyzes the results of the simulation experiments for two demand scenarios. Finally, Section 6.8 provides some concluding remarks about the obtained results of this application.

6.1 Introduction

Optimum management and control of traffic in urban networks is an important requirement for city authorities, as they for seek efficient, safe and sustainable transport. In addition, there is an increasingly wide range of demanding objectives for transport policy makers to achieve, such as public transport priority, improved conditions for vulnerable road users, real-time traffic information, emergency and incident management, and restraining traffic in sensitive areas.

As a response to these issues, Urban Traffic Management and Control (UTMC) systems have been introduced in many cities around the world, in order to provide the tools to support efficient and effective network management to meet needs of current and future traffic problems. Fundamentally, UTMC systems are conceived as modular open systems, that incorporate and build on existing functionalities of existing signal control and other

traffic management systems. An important point to note, is that the Urban Traffic Control (UTC) systems are often at the heart of UTMC and provide a better migration path so that improvements in UTC are utilized to the full in UTMC.

UTC refers to the control of traffic in urban areas using traffic signals, which are linked to operate in a coordinated way. Such linked signal systems may be used to achieve a variety of policy objectives, which relate to efficiency of traffic operations, improved safety, reduced atmospheric pollution, priority for specific road user groups, access control to maintain or enhance urban environments, and to mitigate the effects of irregular events such as accidents or road closure. UTC systems use historic or (more commonly) real-time knowledge of network conditions in order to determine the most appropriate control strategy, and signal infrastructure in order to inform and control the road users.

Early UTC systems in the 1950s and 1960s were based on fixed-time traffic control, providing signal coordination or progression for traffic on an arterial, through the optimization of offsets between adjacent sets of signals. UTC, was therefore justified on there being a sufficient density of traffic signals to make signal coordination worthwhile, compared to the alternative of operating traffic signals in isolation. Whilst relatively effective for traffic coordination in "predictable" conditions, the inability of fixed-time systems to adjust to changing traffic conditions has been a drawback in this approach. The desire for traffic signaling to be more responsive to changing traffic conditions has led to the development of a range of semi or fully traffic responsive UTC systems. The improved performance of these systems has generally justified their additional cost (e.g. detection, maintenance, etc.).

A variety of methods for UTC have evolved over the last decades, responding to the needs of individual cities/countries, the existing research and development base and advances in detection, communications and control technology. These traffic-responsive UTC systems are continuously upgraded to meet with current requirements. Quality attributes of a UTC system play a major role in its architecture. These may include attributes such as the speed of system response to recurring congestion and incidents (i.e. responsiveness), feedback philosophy, ability of integration, functional and spatial extendability, wider range of control strategies, robustness, installation and maintenance costs, etc. Flexibility of the system to incorporate enhancements as policies/technologies advance is a further key attribute.

Criteria for installing a real-time UTC system are now much wider than the need for efficient signal coordination for traffic. For example, the UTC communications infrastructure and processing capabilities give a powerful tool for the network manager, including such functions as traffic information, automatic incident detection (AID), and congestion management.

6.2 Famous UTC systems

UTC systems constitute a scientific field with long-lasting and extensive research and development activities. Many methodologies have been proposed so far, but there is still space for new developments, particularly for saturated traffic conditions. In fact, widely used strategies like SCOOT [29] and SCATS [48], although applicable to large-scale networks, have been judged to have a limited traffic-responsive behavior during rapidly changing conditions, because of accordingly limited incremental changes of the signal settings and their decentralized functional architecture.

On the other hand, more advanced traffic-responsive strategies like OPAC [21], PRODYN [18], and RHODES [55] use algorithms with exponential complexity, which do not permit a straightforward central network-wide application. As a consequence, they use heuristic, hierarchically superior control levels with the aim of network-wide coordination. Finally, most available strategies face modeling limitations when it comes to saturated traffic conditions that are frequently observed in modern metropolitan areas.

6.3 The UTC strategy TUC

In contrast to the aforementioned control strategies, TUC (Traffic-responsive Urban Control) has been developed to provide coordinated, network-wide, traffic-responsive control in large-scale urban networks, especially in cases of saturated traffic conditions. This objective is approached by using appropriate methodological tools, that allow efficient application to large-scale networks and give rise to the following characteristics:

- High efficiency as demonstrated by a variety of results under both simulated and real-life traffic conditions.
- Robustness with respect to measurement inaccuracies, disturbances, and hardware failures (detectors, communication links, etc.).
- Generality that leads to easy applicability (via available software tools) in networks of arbitrary characteristics and dimensions.
- Extreme simplicity of design and implementation code.
- Limited measurement requirements (one detector per link, arbitrary detector location within links) and communication requirements (measurements or decisions once per cycle).
- Low computational effort.

TUC was initially developed as part of an integrated traffic control system for corridor networks within the European Telematics Applications in Transport project TABASCO (Telematics Applications in BAvaria, SCotland and Others). The first version of TUC was controlling only the green splits; after the initial development and the first successful field implementations and evaluations, TUC was expanded to perform real-time cycle and offset control too. Within the European Information Society Technology project SMART NETS (Signal MAnagement in Real-time for urban Traffic NETworkS), the cycle and offset control extensions of TUC have been further investigated, while a new extension was introduced to allow for public transport priority. The extended strategy was field-implemented and evaluated in three European cities (Southampton, United Kingdom; Munich, Germany; Chania, Greece; see [37] for details) within the frame of SMART NETS, while several simulation tests using the microscopic simulator AIMSUN [1] have been conducted in the meantime for parts of the urban networks of Tel Aviv and Jerusalem in Israel. Quite recently, TUC has been applied in several Brazilian cities for network-wide centralized control.

The TUC strategy consists of four main parts (see Figure 6.1):

- **Split control**. This was the first part to be developed. The control objective is minimization of the risk of oversaturation and queue spillback. This control objective is approached through the appropriate manipulation of the green splits at signalized junctions for given cycle times and offsets. The methodology used in this part of TUC strategy is based on the Linear-Quadratic (LQ) regulator theory of automatic control, which is applied network-widely.

- **Cycle control**. It is effectuated through a simple, feedback-based algorithm (P-regulator) that modifies the network cycle time. The control objective is to adapt the cycle duration to the currently observed maximum saturation level in the network.

- **Offset control**. It is effectuated through application of a decentralized feedback control law. The feedback regulator modifies the offsets of the main stages of successive junctions along arterials, to create "green waves", taking into account the possible existence of vehicle queues.

- **Public transport priority**. This part of the strategy is aimed at providing priority to public transport vehicles by modifying locally, in a suitable way, the network-wide signal settings of the previous modules. It was the last part of TUC to be developed.

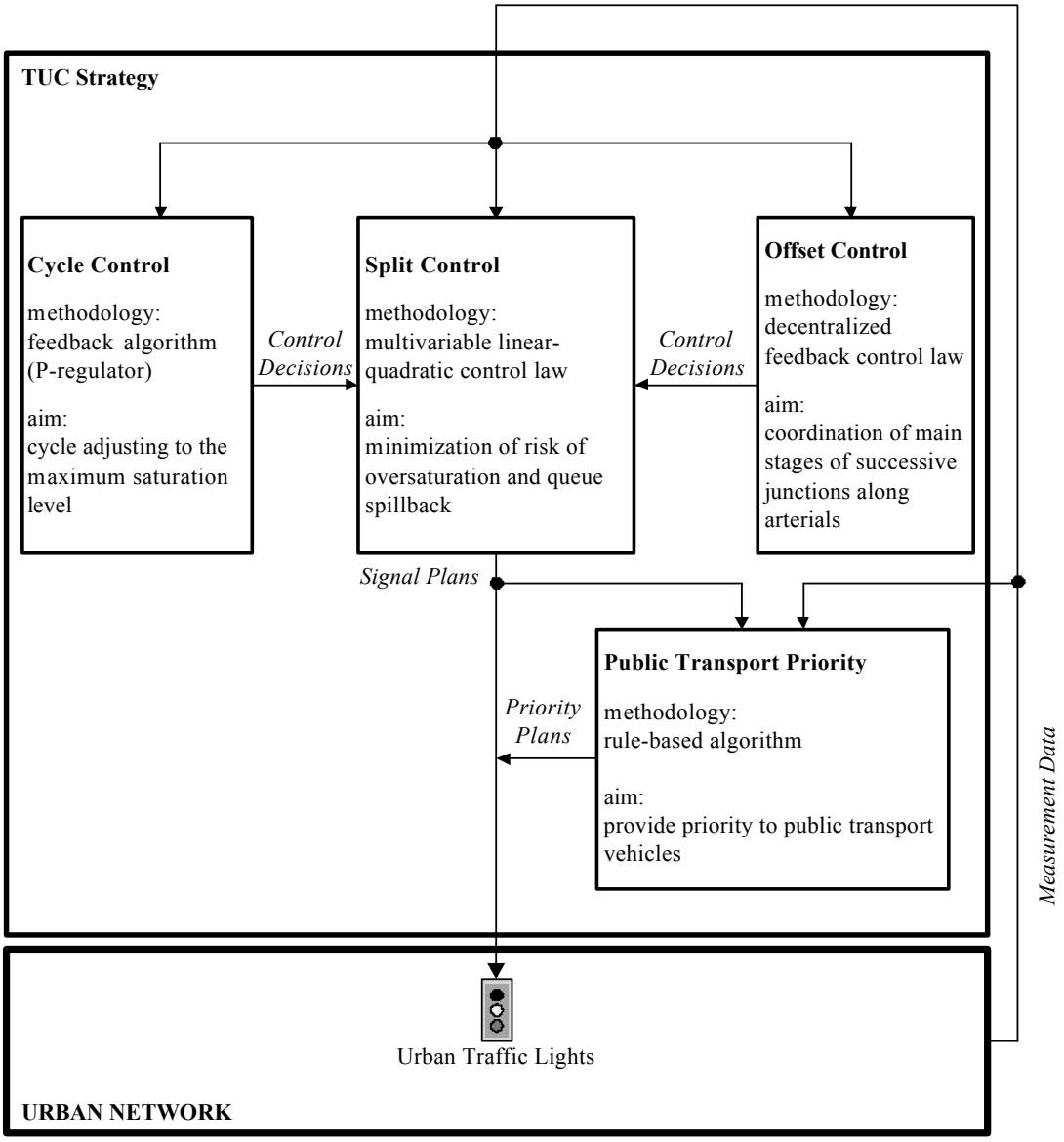

Figure 6.1: Functional architecture of TUC strategy.

It should be noted, that depending on specific user requirements, any combination of the four control parts may be selected for application. For example, the user can select to perform only split control, or split and cycle control, and so forth. Moreover, each part of TUC may run with a different control interval.

As indicated in Figure 6.1, the available real-time measurements from the network are fed (once at each cycle) to the cycle, offset, and split control parts of TUC. Based on these measurements, the cycle and offset control parts perform their control tasks and forward their decisions to the split control part. Given these control decisions and the measurements, the split part then performs its own control task and provides complete network-wide, traffic-responsive signal settings to be forwarded for implementation. If the public transport priority part is also active, it may modify these signal settings locally, in order to serve priority requests (public transport data).

The measurements required by the cycle, split, and offset parts of TUC in real-time are average numbers of vehicles within network links over a cycle. Unless the controlled network is equipped with a video detection system, such measurements are not available. In this case, occupancy measurements collected via traditional loop detectors may be used to estimate numbers of vehicles within links, via suitable nonlinear functions. In addition to the aforementioned measurements, the public transport priority part of TUC uses data related to the movement of public transport vehicles within the controlled network. These data include at least information on the presence of a public transport vehicle within a street.

Finally, a recent extension of TUC strategy [43], uses the real-time flow measurements of loop detectors in order to provide better performance under undersaturated traffic conditions. The new extension and the split control module of TUC are combined in a real-time hybrid control scheme by the implementation of a simple switching logic.

6.3.1 Split control module

The basic methodology used by TUC for split control, is the formulation of the urban traffic control problem as a LQ optimal control problem, based on a store-and-forward type of mathematical modeling. This type of modeling of traffic networks was first suggested by Gazis and Potts in [23] and has since been used in various works, notably for urban traffic signal control. This modeling philosophy circumvents the inclusion of discrete variables in the signal control problem formulation, thus allowing the application of polynomial-complexity solution methods of optimization and control. This part is been described in details elsewhere ([13], [12], [10]), hence, a short summary is provided here.

The control objective is to minimize the risk of oversaturation and the spillback of link queues by suitably varying, in a coordinated manner, the green-phase durations of all stages at

all network junctions around some nominal values, without affecting the offsets or cycle times. The LQ approach leads in a straightforward way to the following multivariable regulator (see [13], [12], [10] for details):

$$g(t_s) = g^{\mathrm{N}} - Lx(t_s) \tag{6.1}$$

where:

- $t_s = 0, 1, 2, \ldots$ is the discrete time index reflecting corresponding signal cycles.
- g is the vector of the green times of all stages and all junctions in the network; $g(t_s)$ are the green times to be applied during the starting cycle t_s.
- g^{N} is the vector of nominal green times for all network stages; these nominal green times correspond to a prespecified fixed signal plan for the network.
- x is the vector of the vehicle-numbers in all network links; $x(t_s)$ are the vehicle-numbers at the start of cycle t_s, i.e. at the end of the previous cycle $t_s - 1$; thus $x(t_s)$ represents a feedback from the network under control, based on which, the new green times are calculated via equation (6.1) in real-time.
- L is a constant gain matrix (of appropriate dimensions) that is calculated off-line based on a straightforward procedure according to the LQ regulator methodology. The matrix depends on the network geometry, the turning rates and the saturation flows, but was found to be little sensitive to moderate variations of these values [10], [2].

Calculation of L is the straightforward outcome of the LQ problem formulation and is carried out off-line (at the lab) once per application network, while the online real-time calculations are limited to the execution of equation (6.1) with a given constant control matrix L and state measurements $x(t_s)$. After the application of equation (6.1), a simple low-cost algorithm ([10]) applies any existing constraints (e.g. cycle constraints and minimum admissible green times) to the obtained values of $g(t_s)$. Given the split decisions of this part of the TUC strategy, as well as the input this part has received from the cycle and offset control parts, complete network-wide signal settings including cycle, split, and built-in offsets are available for implementation at the end of split control.

6.3.2 Cycle control module

One way to influence traffic conditions via traffic lights is by modifying cycle time. Note that one single cycle time is considered here for the whole network, in order to enable junction coordination via suitable offsets. Fundamentally, a longer cycle time typically increases the

junction capacity, because the proportion of the constant lost (intergreen) times becomes accordingly smaller; on the other hand, a longer cycle time may increase vehicle delays in undersaturated junctions due to longer waiting times during the red phase.

Considering the aforementioned remarks, the objective of cycle control should be to increase the junctions capacities as much as necessary to limit the maximum observed saturation level in the network. Within TUC, this objective is effectuated by application of a simple feedback algorithm, that uses as a criterion for the increase or decrease of the cycle the current maximum saturation level of a prespecified percentage of the network links ([11]).

The feedback algorithm for cycle control comprises three steps:

1. A prespecified percentage of network links with currently maximum load (link load $\sigma_z(t_c)$ for link z derives from $\sigma_z(t_c) = x_z(t_c)/x_{z,\max}$, with $x_z(t_c)$ the vehicle-number in the link and $x_{z,\max}$ the capacity of the link) are identified and the corresponding loads are averaged to provide the average maximum load $\sigma(t_c)$.

2. The network cycle $C(t_c)$ is calculated from the feedback control law (P-regulator)

$$C(t_c) = C^{\mathrm{N}} + K^{\mathrm{c}} \left[\sigma(t_c) - \sigma^{\mathrm{N}}\right] \tag{6.2}$$

   where:

   - $t_c = 0, 1, 2, \ldots$ is the discrete time index for applying cycle control.
   - C^{N} a nominal network cycle time (e.g. equal to the minimum admissible cycle $C_{\min}$).
   - σ^{N} a nominal average load (e.g. equal to zero).
   - K^{c} a tunable control parameter, the value of which affects the intensity of the control reactions.

   After applying equation (6.2), the calculated cycle time is constrained within the range $[C_{\min}, C_{\max}]$, if necessary, to become feasible, where $C_{\min}$ and $C_{\max}$ are the minimum and maximum admissible network cycle times, respectively.

3. If the resulting network cycle time $C(t_c)$ is sufficiently high, while all links approaching specific junctions have sufficiently low saturation levels – that is, their current load is less than a prespecified threshold – then these undersaturated junctions are double cycling (i.e. they run cycle times equal to $C(t_c)/2$).

The first two steps described above attempt to adjust the network cycle time to the observed maximum saturation level, while the third step attempts to reduce the delays that

would occur at specific undersaturated junctions because of high cycle times. The control-theoretic justification for the P-regulator of equation (6.2) is in accordance with the mentioned impact of the cycle time on the junctions capacities. More precisely, a higher cycle time increases proportionally the junction capacity; the latter, on its turn, leads to a continuous reduction of the link loads, which corresponds to an integrating impact. In other words, the process under cycle control includes an integrator; hence, a P-regulator is perfectly suitable to stabilize the link load around a desired value (zero).

The cycle time specified according to the previously described logic is then forwarded to the split control part to be used at the next split control interval in determining green splits for the considered junctions.

6.3.3 Offset control module

Another way to influence the traffic conditions, is by specifying the offset between successive junctions. This way it is possible to create a "green wave" along an arterial. The specification of offset, ideally should take into account the possible existence of vehicle queues (particularly under saturated conditions), and this is in fact attempted by the offset control part within TUC.

The offset control module of TUC ([11]) is based on the following assumptions:

- Offset is initially specified along one-directional arterials that do not intersect. Note that arterials are defined here as an arbitrary sequence of links that do not need to correspond to physical network arterials.

- In the case of two-directional arterials, an offset is specified for each direction and the offset that finally will be implemented is a weighted mean of the offsets of the two directions. Alternatively, the most loaded direction may be selected (in real-time) to determine the arterial offsets.

- In the case of arterials that do intersect, TUC considers a prespecified priority order according to their relative importance regarding offset specification; then offset control is implemented to each arterial sequentially, starting from the arterial that has highest priority.

TUC performs offset control in a decentralized way, that is, for successive couples of junctions along the predefined offset arterials. For each couple of junctions, the offset specification changes the starting time of a specific **main stage** of the upstream junction, whereby this main stage is uniquely determined from the arterial composition. TUC considers the possi-

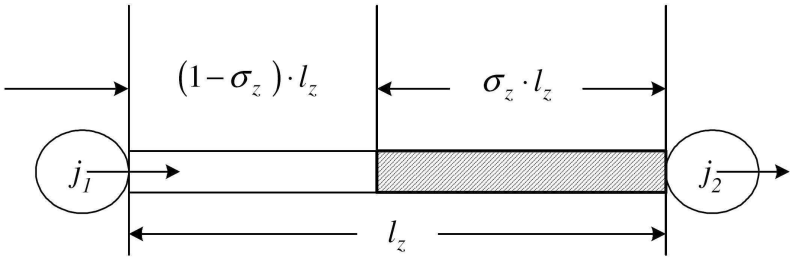

Figure 6.2: Offset calculation between two successive junctions j_1, j_2.

ble existence of vehicle queues while specifying the offset between two successive junctions, through the application of a simple decentralized feedback control law.

Consider two successive junctions j_1 and j_2 and the network link z that connects them leads from j_1 to j_2 and receives right-of-way during the main stage of junction j_2 (Figure 6.2) with l_z the link length and ν_z the free-flow mean speed on link z. Note that the queue length of link z (gray part in Figure 6.2) is approximately equal to $\sigma_z(t_c)\cdot l_z$ (recall that $\sigma_z(t_c) = x_z(t_c)/x_{z,\max}$). If there are no vehicles in the link, the offset between the two intersections should be equal to the travel time under average speed, that is, l_z/ν_z. In other words, the cycle in j_2 should start after the cycle in j_1 (positive offset). As the number of vehicles in link z grows, the offset should decrease correspondingly in order to allow the partial discharge of the queue in j_2. Then, the cycle in the downstream intersection should begin earlier than in the no-queue case and, in some cases, even before the cycle in the upstream intersection (negative offset).

More specifically, an ideal offset would be obtained, if the following two traffic flow waves meet exactly at the tail of the existing queue:

1. Flow wave created due to the switching to green light at the upstream junction j_1; this wave moves downstream with speed ν_z, hence, it will reach the queue tail at time $[1-\sigma_z(t_c)]\cdot l_z/\nu_z$ after the green switch.

2. Kinematic wave created due to the switching to green light at the downstream junction j_2; this wave moves upstream (along the queue) with speed ν^c, which generally is estimated to be about 15 km/h. Hence, the kinematic wave will reach the queue tail at time $\sigma_z(t_c)\cdot l_z/\nu^c$.

Based on the preceding, the ideal offset $t_{j_1,j_2}(t_c)$ in the direction that leads from j_1 to j_2 should satisfy

$$\frac{[1-\sigma_z(t_c)]\,l_z}{\nu_z} = t_{j_1,j_2}(t_c) + \frac{\sigma_z(t_c)\cdot l_z}{\nu^c}. \tag{6.3}$$

Solving this equation for $t_{j_1,j_2}(t_c)$, one obtains the offset feedback control law

$$t_{j_1,j_2}(t_c) = \frac{l_z}{\nu_z} - l_z K_z^o \frac{x_z(t_c)}{x_{z,\max}} \tag{6.4}$$

where $K_z^o = (\nu^c - \nu_z)/\nu_z\nu^c$ is a tunable control parameter. Note that equation (6.4) apparently was derived several times independently, first by Gazis [22], but also in [3] where it eventually was used in a different way. Generalization of the previously described logic leads to the relevant formulas which are used at each successive couple of junctions (j_i, j_{i+1}) along an arterial, in order to specify the offset between the two junctions and consequently along the arterial.

In order to implement the new offset specified in equation (6.4), a transient cycle time C_{j_2} is temporarily implemented in junction j_2. The transient cycle is implemented one single time, after which, the junctions are coordinated according to the new offset. The transient cycle times specified from the offset control part of TUC, as described, are forwarded to the split control part to be used at the next split control interval for determination of green splits of the considered junctions. In case the cycle control part of TUC delivers simultaneously a new cycle, the transient cycle is used first one single time to implement the new offsets, after which the cycle delivered by the cycle control part is implemented until a new transient cycle is produced.

6.3.4 Public transport priority control module

There are two approaches through which TUC may provide priority to public transport vehicles (PTVs):

- Appropriate weighting of the measurements used in the split control law to reflect the presence of PTVs.
- Implementation of an additional module that modifies locally (i.e. at each junction) TUCs network-wide decisions to provide priority to PTVs.

The first approach is suitable for networks with many partially crossing public transport lines and frequent movements of PTVs. Its implementation is easy and it actually forces the split decision algorithm of TUC to favor the movements of PTVs; it does not provide priority in the classical sense of the direct (or at least the sooner possible) switching of the traffic lights to allow a detected PTV to pass. However, it has the advantage over the second approach that it avoids creating major disturbances to the signal plans. To apply the first approach, TUC merely needs to know the number of PTVs within the network links. This

number is used to appropriately weight the measurements used by TUC, thus, forcing the split decision process to favor the links that have major movements of PTVs.

The second approach provides priority in the classical sense of the direct modification of the signal state in the presence of PTVs. However, it is not suitable for networks with many, partially crossing public transport lines and frequent movements of PTVs. This is mainly because it would be hardly possible to modify the signal state as many times as necessary to serve frequently received priority requests.

According to the second approach, if a PTV is detected within a link, the state of the traffic signal is directly modified to allow the vehicle to cross the junction at the earliest possible time. In general, modifying the normal signal state to provide priority to special vehicles may include green extension, stage recall, stage reordering, insertion of a special stage, stage skipping, and so forth. In TUC, priority is provided only by green extension or stage recall because the other methods are hardly justifiable in the context of providing priority to PTVs in the case of saturated traffic conditions.

Modifying the signal plans to provide priority to PTVs is based on the estimated time required for the vehicle to travel from the detection location to the stop-line. In the case of dedicated (bus or light rail train (LRT)) lanes, this travel time is readily calculated based on the PTVs nominal speed. In the case of mixed traffic, the estimation is based on the discharge rate of vehicles that are estimated to be present between the detection location and the stop-line.

The public transport priority part of TUC executes a sequence of steps that in general terms include:

1. The strategy checks whether the criteria for granting priority are satisfied. If the criteria are satisfied, the strategy proceeds to decide whether priority will be provided within the current or a next cycle or whether a priority will be declined.

2. In the case that a request will be considered, the strategy uses the time that the priority request was received in combination with the time the priority settings will be delivered to the junction controller (consideration of transmission delays), to find whether the reply to this particular request refers to

   - a stage that will have been completed,
   - a stage that will not have started, or
   - the stage that will be active.

3. In each of these three cases, the strategy selects the most appropriate way to reply between no action, green extension, and stage recall via minimization of all intermediary

stages.

For further details about the application of TUC's public transport priority control module the reader is referred to [11] and [14], as it is not of primary interest for this thesis. The AFT algorithm described in Chapter 4 is applied for fine-tuning to the split, cycle and offset control modules of strategy TUC.

6.3.5 Traffic measurements

To apply split, cycle, and offset control as well as provision of public transport priority with TUC, availability of measurements of numbers of vehicles in all links is required in real-time. Since the control intervals of TUC are relatively long (e.g. at least equal to the network cycle), these measurements should reflect the (most recent) average traffic conditions and not the periodic fluctuations due to e.g. the green/red switching of the traffic lights. To this end:

1. The numbers of vehicles utilized by the split control part of TUC are the mean values of the corresponding measurements over the previous split control interval.

2. The numbers of vehicles utilized by the offset control as well as the public transport priority part of TUC are the numbers of vehicles used by the split control part during the last control period; however, for the cycle control module the mean values of the corresponding measurements over the previous cycle control interval are used.

This way, all modules of TUC base their control decisions on the most recently observed average traffic conditions in the network.

If an advanced video detection system (video sensor) is available, the required numbers of vehicles within links can be directly collected. Otherwise, the required measurements are estimated based on local occupancy measurements, collected in real time by traditional detector loops.

6.4 The microscopic simulator AIMSUN

This section provides a short introduction to the commercial microscopic traffic simulator AIMSUN. The reader is referenced to [1] for further details.

In general, microscopic modeling of traffic flow considers the movement of individual vehicles in dependence of the movement of adjacent vehicles. Although the basic kernel of this description is expressed in form of differential equations (car-following model), there are

a series of additional actions, like, for example, lane choice, lane change, merging at on-ramp and off-ramp areas, that can only be described via heuristic rules.

Most micro-simulation models use various algorithms and driver behavior models to simulate the movement of individual vehicles within a network. Each vehicle that enters the network is assigned a vehicle type (auto, truck, bus, etc.) and corresponding vehicle performance characteristics (acceleration, deceleration, speed, and turning characteristics). It is also assigned one of ten driver characteristics (ranging from aggressive to cautious), giving each vehicle a unique and realistic performance profile that it maintains while traveling through the network. The position and speed of each vehicle on the network is updated once per simulation step based on its own performance and driver characteristics, the actions of vehicles around it, roadway properties, and traffic control devices. Thus, the interaction of vehicle to vehicle, vehicle to road, and vehicle to control devices are modeled accurately for each simulation. Default vehicle and driver characteristics can also be modified to better reflect actual traffic conditions for a given scenario.

Once a vehicle is assigned performance and driver characteristics, its movement through the network is determined by three primary algorithms: (a) car following algorithm, (b) lane changing algorithm and (c) gap acceptance algorithm. There are other algorithms which influence vehicle behavior too, such as those which govern queue discharge and traffic signal control, but these three are perhaps the most important and are common to all traffic simulation models.

AIMSUN (Advanced Interactive Micro-Simulation for Urban and non-urban Networks) is a full function microscopic simulation tool with a broad range of simulation capabilities. It can simulate surface street networks, motorways, interchanges, weaving sections, pretimed and actuated signals, stop controlled intersections, and roundabouts. It also includes features about full trip distribution capabilities, dynamic traffic assignment, real-time vehicle guidance, and 3-D animations. AIMSUN is used in conjunction with the traffic network graphical editor (TEDI) and is part of the Generic Environment for Traffic Analysis and Modeling (GETRAM) open simulation environment.

Vehicle attributes such as length, width, maximum speed, and normal and maximum acceleration are assigned when a vehicle enters the network. Users can select from a wide variety of vehicle types, and within each type there will be some variation in these parameters based on statistical distributions. Within the vehicle stream there is variation of driver performance characteristics, such as desired minimum headway, turning speed and speed acceptance (obedience to the speed limit). AIMSUN establishes mean driver performance values and varies driver behavior for each vehicle about the mean (within specified minimum and maximum values).

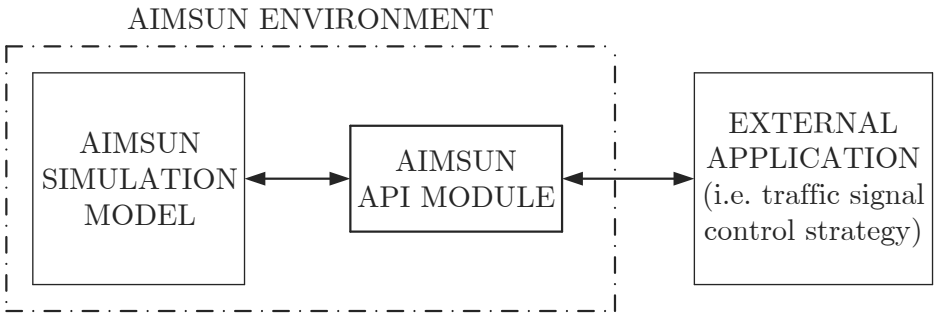

Figure 6.3: Schematic representation of the AIMSUN API module.

AIMSUN can function as either a stochastic model, where vehicles travel through the network based on turn probabilities, or a traffic assignment model using O/D matrices. It is also capable of providing dynamic traffic assignment, where optimum vehicle paths between centroids are computed at the beginning of the simulation and then updated based on feedback from the network. Thus, route choice is based on actual traffic conditions and may vary at different points during the simulation horizon. Finally, AIMSUN can simulate the effects of ITS (Intelligent Transportation Systems), providing active vehicle guidance (either variable message signs or in-vehicle systems) to modify route choice during a simulated incident.

The evaluation of different UTC strategies is achieved in AIMSUN by the use of API (Application Programming Interface). The AIMSUN API module extends the functions of AIMSUN environment including user defined applications, which can exchange information (e.g. traffic state of the network) and/or modify its state dynamically (e.g. via the change of traffic signal plans). The AIMSUN API module (Figure 6.3) is placed, in the functional point of view, between the AIMSUN simulation model and the external application defined by the user (e.g. real-time UTC strategy).

There are two types of communication processes: on one hand there is the communication process between AIMSUN and AIMSUN API module and on the other hand between the AIMSUN API module and the external application. The communication process between AIMSUN API module and AIMSUN simulation model is provided by the AIMSUN environment; albeit, the communication between the AIMSUN API module and the external application has to be implemented by the user, depending on the requirements of the external application. For the simulation experiments of this thesis, TUC strategy was implemented as an external application and connected with AIMSUN using the API module.

6.5 The urban road network of Chania

Chania, located at the north-western part of Crete, is the capital of the prefecture of Chania and covers 12.5km^2. Chania is the second biggest prefecture of Crete in size, population and development. Figure 6.4 exhibits a satellite view of the trial urban road network (red bullets correspond to the controlled junctions), which has a total length of approximately 8km and consists of 16 controlled junctions. Most of the links in that network consist of only one lane, which means that unexpected events (accidents, double-parking, etc.) may block the link and therefore deteriorate the traffic conditions, even if their duration is only a few minutes.

Moreover, congestion problems are not limited in the streets with the unexpected events but they are propagated to many other streets and may sometimes lead to partial grid-lock situations. Thus, the implemented control strategy should be able to deal with those problems. During the morning and evening hours there is a frequent bus service in almost every part of the network. Pedestrian movements are not a problem in the network and there is no reason for a special treatment. Public transport priority is not a subject in Chania, so it is not implemented in the experiments.

Congestion problems are encountered every day especially in the central (Chatzimichali Gianari, Kidonias, Apokoronou) and the northern part (Manousou Koundourou, El. Venizelou) of the network for about one to two hours in the morning and evening. In most traffic arterials of the city there is heavy congestion 19:00–21:30 on Tuesday, Thursday and Friday evening, because of the shopping centre. Another reason for the congestion is the high frequency of buses, which embark and disembark people at stops frequently blocking one direction of the street. Other reasons are the lower capacity due to illegal parking on the main streets and the high usage of vehicles by the residents of the city.

Heavy congestion problems are emerging on the entire network during the rainy days when there is an excessive inner and outer demand, usually a demand that cannot be sustained by the network's infrastructure. Heavy congestion problems are emerging, also, during the tourist summer season. The heavily loaded urban network of the city is further loaded by additional private cars and motorbikes, rented cars and bikes, tourist buses, and by the increased movement of taxis. The aforementioned problems are encountered every day except Sunday and some rare occasions such as off-days. The city of Chania suffers from traffic congestion, lack of parking supply and traffic-generated pollution of the natural and built environment.

Figure 6.4: Satellite view of the Chania urban road network.

6.6 Application of AFT to TUC strategy

In order to evaluate the efficiency of AFT algorithm presented in Chapter 4 to the problem of optimizing the design parameters of the split, cycle and offset control modules of TUC, extensive simulation experiments have been carried out, comparing the performance of AFT to the base-case (no AFT case). In the base-case, the aforementioned design parameters of the original TUC system were manually fine-tuned to virtual perfection by the system operators [37].

6.6.1 Network and simulation setup

Figure 6.5 represents the model of the network developed for the simulation investigations. It consists of 16 signalized junctions (nodes) and 60 links (arrows). Each network link corresponds to a particular junction phase. Typical loop-detector locations within Chania urban network links are either around the middle of the link or some 40m upstream of the stop-line. The traffic network characteristics (turning rates, lost times, staging, and saturation flows) and the fixed plan g^{N} of equation (6.1) used in AIMSUN and in TUC were provided by the system operators of the Traffic Control Centre (TCC) of the city. Note that the fixed plan g^{N} is one of the six fixed predefined network signal plans used by the TCC. Split, cycle and offset control modules of TUC strategy are applied to the network for all simulation investigations.

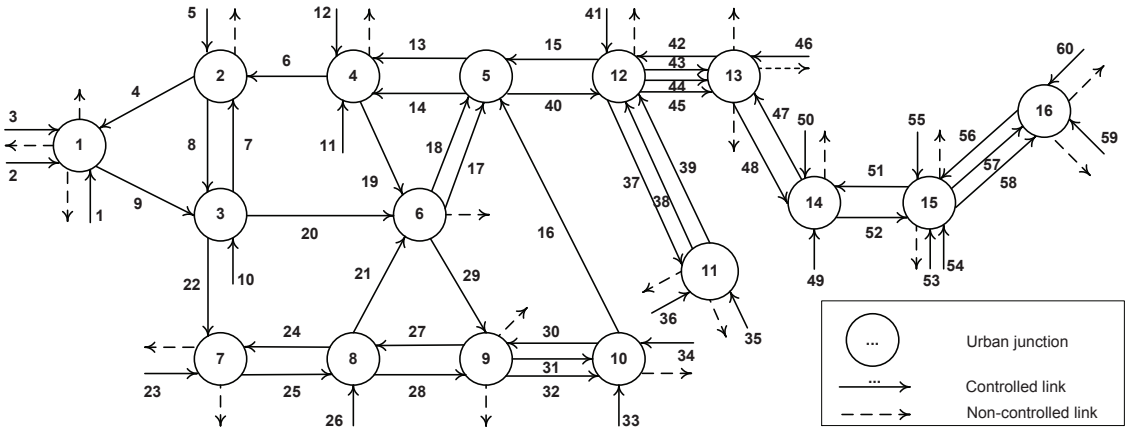

Figure 6.5: Simulation model of the Chania urban road network.

Finally, a simulation step of 0.25s is considered for the microscopic simulation model.

For the application of the TUC strategy the following typical design values were used: $k_{\mathrm{s}} = C$, $k_{\mathrm{c}} = 600s$, $C_{\min} = 60s$, $C_{\max} = 120s$, $C^{\mathrm{N}} = C_{\min}$. Also, for the implementation of AFT algorithm the following design values were used: $T_{\mathrm{h}} = 90$, $\bar{L}_{\mathrm{g}} = 150$, $K = 100$ and initial values to λ_i according to $\lambda_1 = 100$, $\lambda_3 = 0.1$, $\lambda_2 = \lambda_4 = 0$.

The exogenous vector $x(k)$ in this application corresponds to the traffic demand (time-history of the number of vehicles entering the network in each of the network origins). Following the same procedure as described in Section 5.5.2, an estimate of the demand $\bar{x}(k)$ every 1h was constructed (i.e., four different estimates for the demand of the whole day). This low-dimension, noisy estimate $\bar{x}(k)$ of the traffic demand was again based on the average number of vehicles entering every particular origin over the 1-h period and resulted in vectors of dimension $n_x = 88$ (4(hours)x22(network origins)).

6.6.2 Tunable parameters

In this section the design parameters of TUC strategy selected for fine-tuning via AFT algorithm are thoroughly presented. The next three paragraphs describe the reasoning for the definition of the set of tunable parameters for the split, cycle and offset control respectively.

In Section 6.3.1 the multivariable regulator of the split control module of TUC was described. The aim of this regulator (equation (6.1)) is to balance the relative space occupancies $x_z/x_{z,\max}$ in the network links, so as to minimize the risk of queue spillovers which may lead to a waste of green time and even to gridlocks. LQ regulator may apply an inherent gating, i.e., reduce the green time of links that feed a saturated road, even if these links are two or more junctions away. The number of vehicles $x_z(t_s)$ for link z during the last cycle t_s is

estimated via the following equation:

$$x_z(t_s) = x_{z,\max} f\left(o_z(t_s), \Lambda_z\right) b_z \tag{6.5}$$

where $o_z(t_s)$ denotes the measured average time-occupancy (measured by loop-detectors located at a certain distance from the stop-line) during the last cycle time; $f(\cdot)$ is an empirical function [12], [37] constructed from practical investigations; Λ_z denotes the distance of the loop-detector from the stop-line divided by the total link length. Finally, b_z is a non-negative design parameter for each link z (so-called "importance factor") that is introduced, such that the $x_z(t_s)$-values resulting from (6.2) are multiplied with the corresponding b_z before being used by the multivariable regulator (6.1). The default values are $b_z = 1, \forall z$, but experienced system operators may manually select a real value $b_z \in (0, 3]$ so as to increase or decrease the importance of specific links, i.e. make them look more or less saturated than the measurements actually reflect. These design parameters are critical for the successful deployment and operation of the signal control strategy TUC, and hence were selected for automated fine-tuning by the AFT algorithm.

In Section 6.3.2 the feedback P-type regulator of the cycle control module of TUC was described. Longer cycle times typically increase the capacity of a junction, but, on the other hand, may increase vehicle delays in undersaturated junctions, due to longer waiting times during the red phase, or, even worse, create queue spillovers. Considering the aforementioned remark, the objective of cycle control module is to increase the junctions' capacities as much as necessary to limit the maximum observed saturation level in the network. For this reason, and for this particular application, equation (6.2) presented in Section 6.3.2 is slightly changed, leading to a new feedback P-type brunch regulator which reads

$$C(t_c) = \begin{cases} C^N + K_1 \left[\sigma(t_c) - \sigma_{N_1}\right], & \text{if } \sigma(t_c) \leq \sigma_{cr} \\ C^N - K_2 \left[\sigma(t_c) - \sigma_{N_2}\right], & \text{if } \sigma(t_c) > \sigma_{cr}. \end{cases} \tag{6.6}$$

The efficiency of the developed regulator is extensively investigated through simulation experiments. $K_1, K_2 > 0$ are network-wide design parameters, the selection of which affects the intensity of the cycle control module reactions, and hence may cause a degradation in the overall performance of TUC strategy if not suitably configured. In other words, high K_1, K_2 values force the control law to react strongly even for small differences of $\sigma(t_c)$ from σ_{N_i}, $i = 1, 2$. For this reason, the design parameters $K_1, K_2, \sigma_{N_1}, \sigma_{N_2}, \sigma_{cr}$ were selected for automated fine-tuning by the AFT algorithm.

Finally, the feedback regulator described in Section 6.3.3 is applied for offset coordination in all simulation experiments. Equation (6.4) is applied to all network links that connect successive junctions that belong to the main arterials of the network. More specifically, in

the urban road network of Chania there are 7 main coordinated arterials: (i) 5–12–13, (ii) 13–14, (iii) 5–4–2–1, (iv) 5–6–3, (v) 14–15–16, (vi) 12–11, (vii) 6–9, which are connected through 12 links: (i) 15–42, (ii) 48, (iii) 13–6–4, (iv) 17–20, (v) 51–56, (vi) 37, (vii) 29 accordingly (Figure 6.5). As mentioned earlier, in (6.4) ν_z denotes the free-flow mean speed on link z and K_z^o a tunable parameter depending on ν_z and ν^c (with ν^c the speed of the kinematic wave moving upstream, along the queue of the link). The parameters K_z^o and $1/\nu_z$ (two for each of the aforementioned links z), which affect the efficiency of the offset control of TUC were selected for automated fine-tuning by the AFT algorithm.

According to the remarks of the three above paragraphs and the notations used in Chapter 4 for AFT algorithm we get $\theta = \mathrm{vec}\,(\theta_1, \theta_2, \theta_3)$; where $\theta_1 = (b_1, b_2, \ldots, b_{60})$, $\theta_2 = (K_1, K_2, \sigma_{N_1}, \sigma_{N_2}, \sigma_{cr})$ and $\theta_3 = (\nu_z, K_z^o), z \in \{4, 6, 13, 15, 17, 20, 29, 37, 42, 48, 51, 56\}$ the design parameters of the split, cycle and offset control modules of TUC, respectively.

The initial values $\theta(0)$ were chosen so as to correspond to values that have been manually fine-tuned in past field implementations of TUC in Chania network. More precisely, parameters θ were initialized according to $\theta_1 = 1$, $\forall z$, $\theta_2 = (240, 300, 0.15, 0.6, 0.4)$ and $\theta_3 = (0.022, 0.04, 0.022, 0.04, 0.033, 0.022, 0.025, 0.022, 0.022, 0.025, 0.028, 0.028, 4.222, 4.933, 4.222, 4.933, 4.666, 4.222, 4.333, 4.222, 4.222, 4.333, 4.476, 4.476)$. In order to assess the overall system performance the criterion $J \equiv ms$ (equation (4.3)) was set to the actual daily network mean speed. Finally, a projection to the nearest valid point is applied whenever a vector θ violates the imposed constraints.

6.6.3 Demand scenarios and integration with AIMSUN

In order to investigate the performance of AFT algorithm under different traffic conditions, two basic traffic demand scenarios (time-history of vehicles entering the network in the network origins during the day) were designed based on actual measurements, each with a simulation horizon of 4 hours. Scenario 1 comprises medium demand in all network origins, while scenario 2 comprises high demand and the network faces serious congestion for some 2 hours, with some link queues spilling back into upstream links. For simplicity, we assume that a demand scenario with a time horizon of 4 hours corresponds to a day. Each day (iteration of the AFT algorithm) a randomly perturbed 5%-width version of the basic demand scenarios is produced and the assessment criterion is gathered from the AIMSUN simulator. Then, the design parameters of TUC strategy are updated by AFT algorithm according to the calculated assessment criterion.

The overall closed-loop scheme consists of two main control loops as inner and outer loops. The inner loop is used by the TUC strategy to produce the traffic signal settings. More specifically at each cycle C, AIMSUN delivers the (emulated) occupancy measurements at

the locations where detectors are placed (as in real conditions). These measurements are used by the control modules of TUC strategy to produce the traffic signal settings (splits, cycle, and offsets). These signal settings are then forwarded to the micro-simulator for application. The outer loop is used by AFT algorithm to update the design parameters of TUC strategy. More specifically, at each day, AIMSUN delivers the mean speed for the whole urban road network. The mean speed is used by AFT algorithm (together with the average demand estimation of next day $\bar{x}(k)$, as defined in Section 6.6.1) in order to produce the new values for the design parameters of split, cycle and offset control modules of TUC strategy (the vector $\theta = \text{vec}\,(\theta_1, \theta_2, \theta_3)$). The new set of the design parameters is then forwarded to TUC strategy for application, and so forth.

6.7 Simulation results

In order to evaluate the efficiency of the presented AFT algorithm to the problem of optimizing the design parameters of split, cycle and offset control modules of TUC, extensive simulation experiments have been conducted. The performance of AFT is compared to the base-case (no AFT case), where the aforementioned design parameters were manually fine-tuned to virtual perfection by the system operators for the original TUC system [37]. AIMSUN is based on stochastic distributions in order to calculate all the internal parameters of the simulations. As a result, two replications of the same simulation are not identical, unless they are fed with the same random seed. For our experiments, 10 simulation runs with different random seeds were carried out for each scenario for statistical justification.

6.7.1 Global results for demand scenario 1

This section presents the simulation results obtained for demand scenario 1 described above (medium demand). Figures 6.6–6.15 compare the network-wide mean speed of the original TUC system (blue line) versus TUC system combined with AFT algorithm (red line) delivered for scenario 1. In all figures, it can be seen that the application of AFT algorithm to the signal control strategy TUC leads to better performance than the original TUC for this demand scenario. More precisely, AFT algorithm achieves to optimize the overall system performance within few days (iteration number in figures), by efficiently fine-tuning the design parameters for all TUC's control modules, while avoiding decreasing the daily mean speed lower than the initial point.

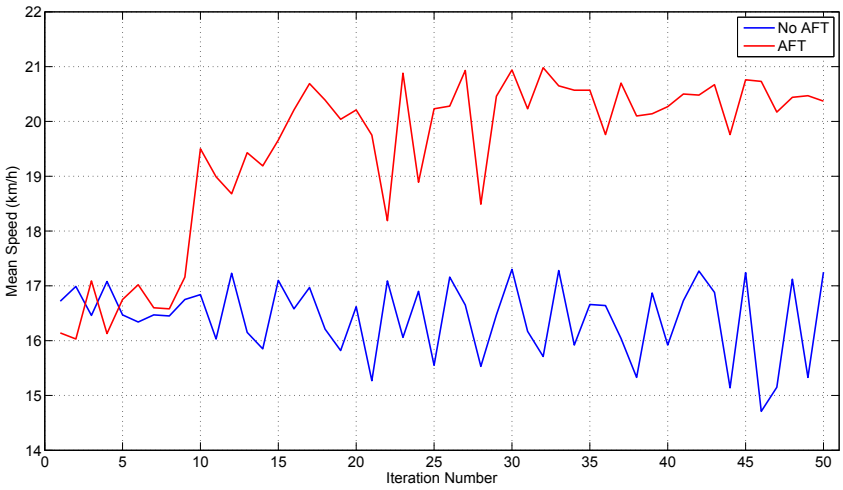

Figure 6.6: Mean speed for demand scenario 1 (TUC/AFT–replication 1).

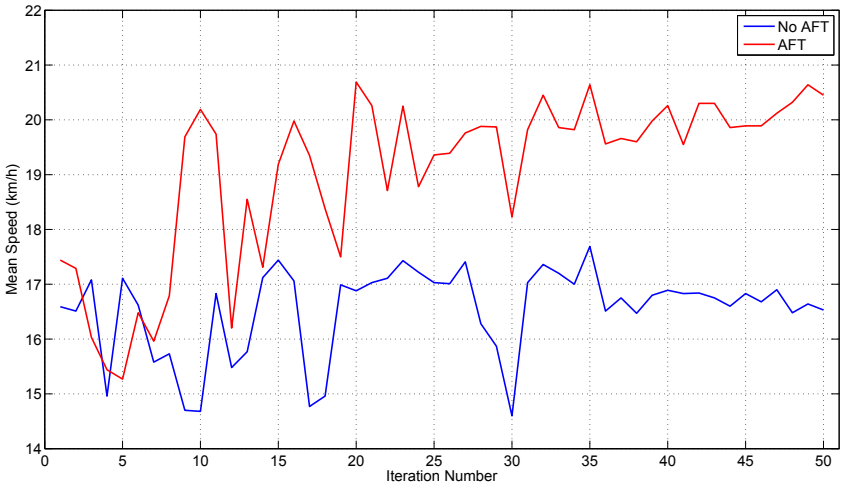

Figure 6.7: Mean speed for demand scenario 1 (TUC/AFT–replication 2).

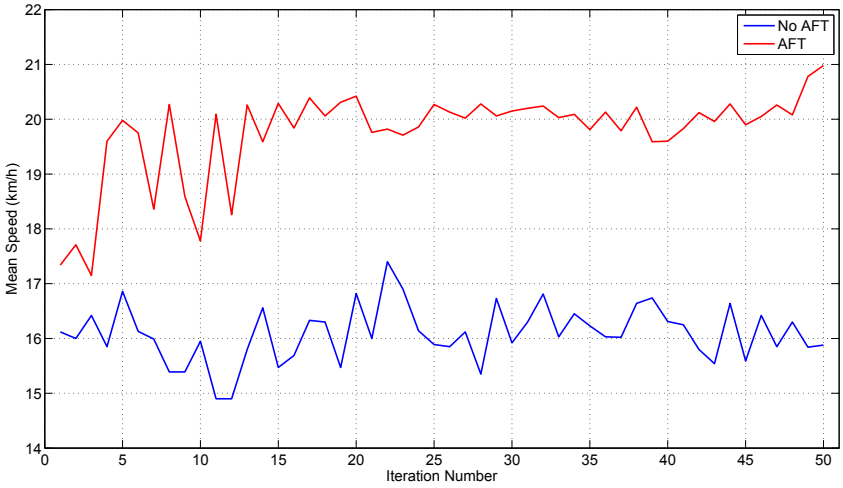

Figure 6.8: Mean speed for demand scenario 1 (TUC/AFT–replication 3).

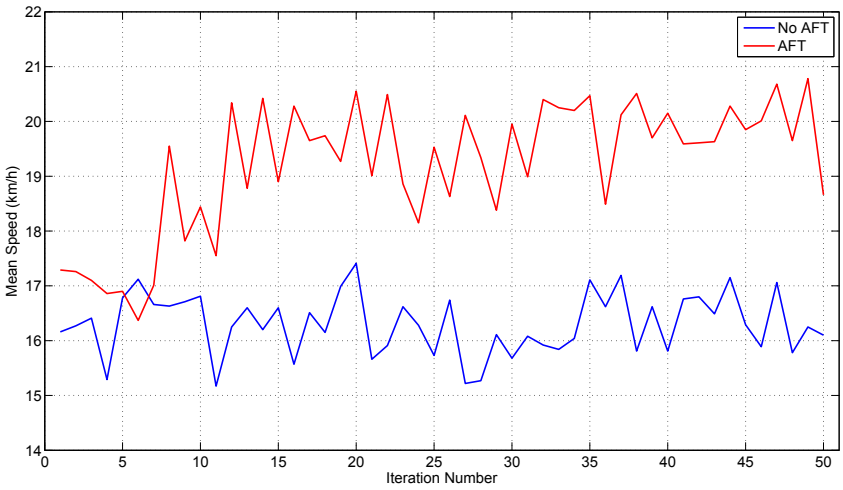

Figure 6.9: Mean speed for demand scenario 1 (TUC/AFT–replication 4).

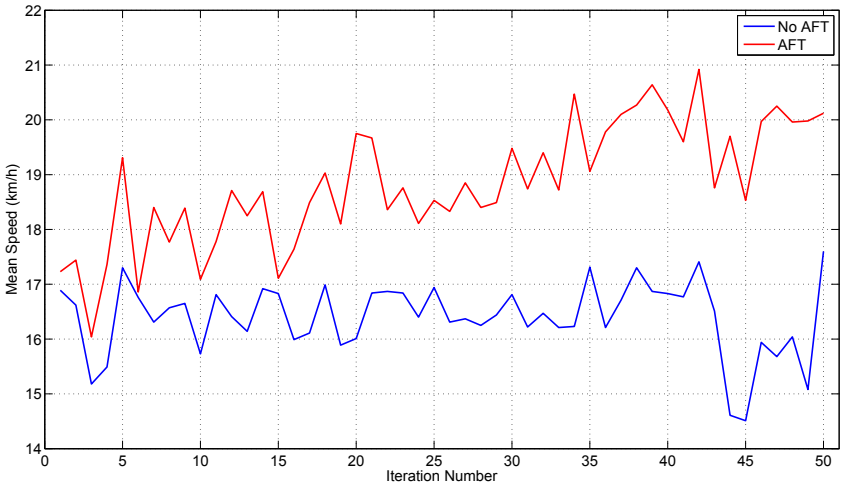

Figure 6.10: Mean speed for demand scenario 1 (TUC/AFT–replication 5).

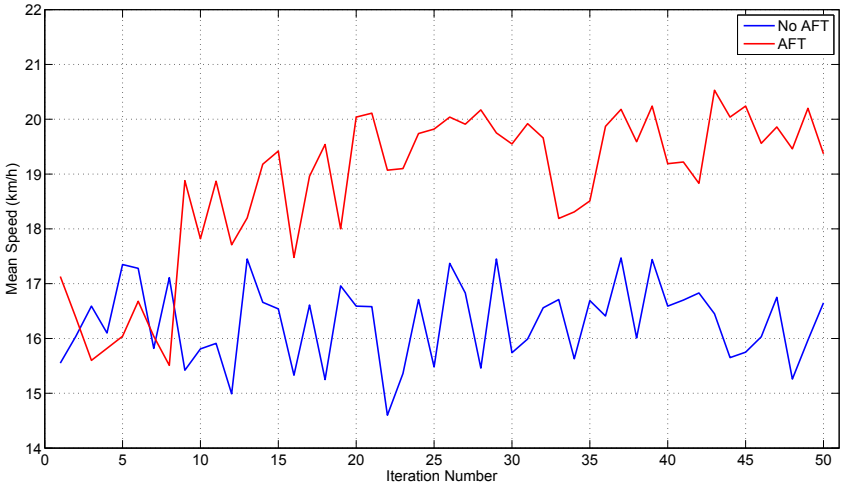

Figure 6.11: Mean speed for demand scenario 1 (TUC/AFT–replication 6).

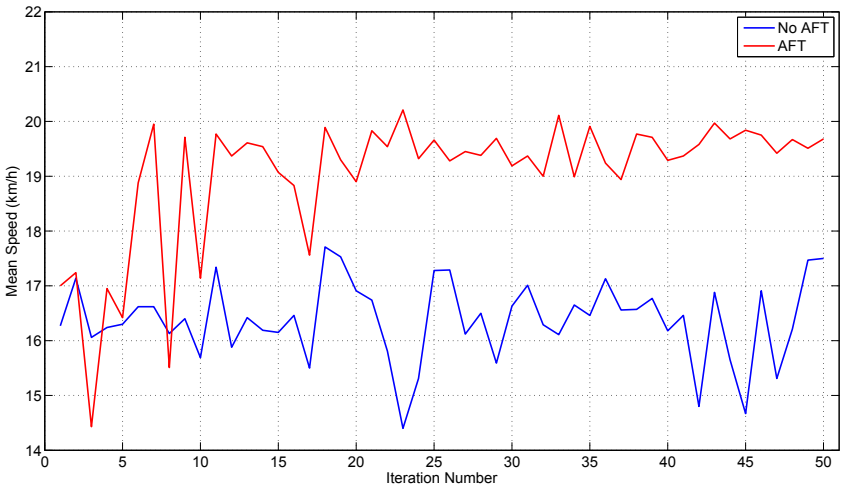

Figure 6.12: Mean speed for demand scenario 1 (TUC/AFT–replication 7).

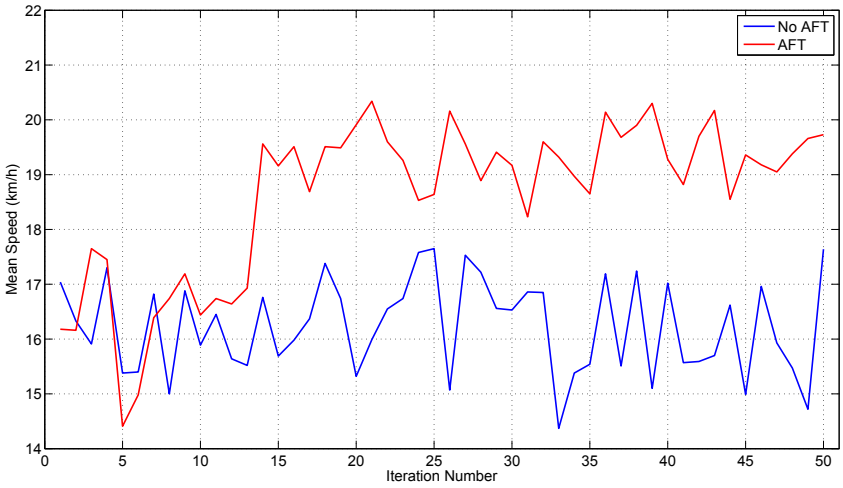

Figure 6.13: Mean speed for demand scenario 1 (TUC/AFT–replication 8).

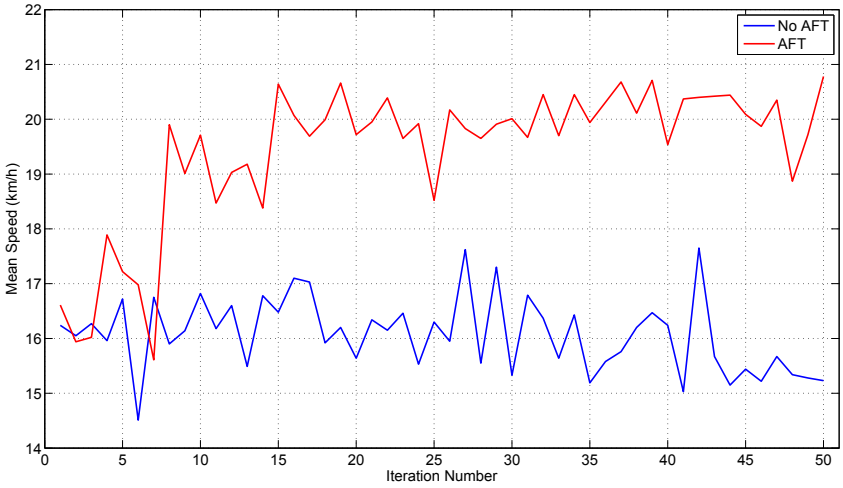

Figure 6.14: Mean speed for demand scenario 1 (TUC/AFT–replication 9).

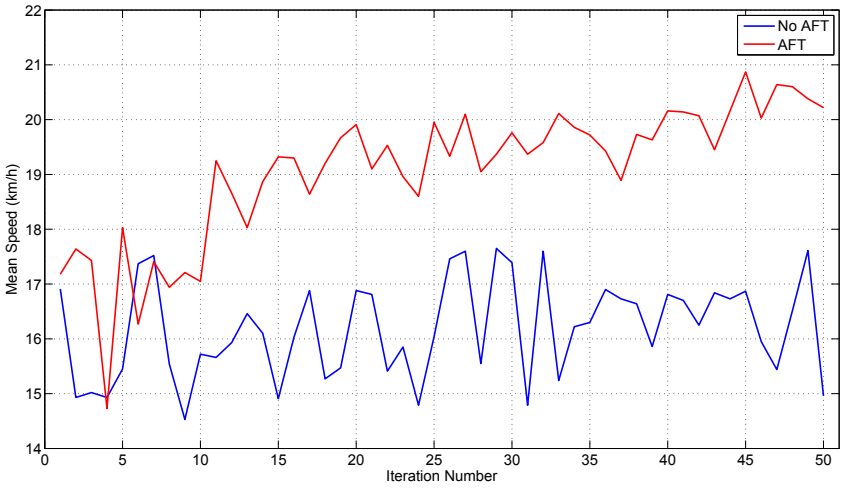

Figure 6.15: Mean speed for demand scenario 1 (TUC/AFT–replication 10).

The trajectory of the system performance (mean speed) is persistently increasing until it converges to a local maximum value. Note that the oscillations appearing in both blue and red lines of the figures are due to the ±5% daily random perturbations applied to the demand scenario. In all simulation runs, equations (4.23)–(4.24) were applied for the stepsize of AFT algorithm, with $\alpha(0) = 1.42$. The descending sequence of $\alpha(k)$ provided by this formula has the advantage that after some iterations descends to small values for the stepsize $\alpha(k)$, providing the algorithm with a solution close to the optimal solution found so far.

Table 6.1 displays the mean speed for the original TUC system and the mean speed when using AFT for the system fine-tuning, for each replication. Also, the standard deviation of the mean speed due to the ±5% daily random perturbations of the demand is presented. It should be noted, that the AFT algorithm is learning the traffic system dynamics during the first iterations (days) and then converges to a local optimal solution (according to equations (4.6) and (5.8)). Thus, the results of AFT are presented for both the whole simulation horizon (column 4 in Table 6.1) and after the convergence of AFT (column 7 in Table 6.1). Finally, Table 6.1 also displays the number of days needed for AFT to converge and the percentage improvement over the no AFT case.

It can be seen that the use of AFT algorithm leads to an average improvement of the system performance of some 17% for this demand scenario; moreover, if the learning period is excluded (which is 8.4 days on average) the improvement increases to some 20%. The variations between different replications due to the stochasticity of the simulator and the random demand perturbations are low. The average mean speed for the original TUC system is 16.29 km/h and after the convergence of AFT this speed is increased to 19.50 km/h.

Table 6.1: Comparison of the average mean speed (MS) with and without the application of AFT algorithm to TUC strategy (demand scenario 1).

	No AFT		AFT (all 50 days)		Conv.	AFT (after convergence day)		
ID	MS	St.Dev.	MS	Improvement	Day	MS	St.Dev.	Improvement
1	16.41	0.68	19.48	18.69%	10	20.04	0.83	22.10%
2	16.53	0.80	19.06	15.28%	9	19.51	1.02	18.02%
3	16.09	0.51	19.76	22.85%	4	19.85	0.71	23.43%
4	16.30	0.56	19.21	17.84%	8	19.52	0.90	19.72%
5	16.40	0.66	18.83	14.80%	5	18.95	1.00	15.55%
6	16.29	0.73	18.83	15.59%	9	19.25	0.97	18.18%
7	16.38	0.73	19.03	16.20%	6	19.26	0.91	17.59%
8	16.23	0.86	18.62	14.73%	14	19.32	0.65	19.02%
9	16.07	0.68	19.43	20.90%	8	19.79	0.88	23.15%
10	16.18	0.90	19.07	17.86%	11	19.53	0.73	20.69%
Average	16.29	0.71	19.13	17.46%	8.4	19.50	0.86	19.73%

For the same simulation experiments the SPSA algorithm (as described in equations (5.9), (5.10)) was applied in order to fine-tune TUC's design parameters. Table 6.2 displays the obtained results for all simulation replications. It is clear that SPSA cannot increase the mean speed of the network. The scale and complexity of the fine-tuning problem prevents equation (5.10) from providing a good estimation of the system's performance gradient. Thus, the applied sets of parameters by SPSA algorithm are "random" and fail to achieve any improvement to the system performance. The average mean speed for the original TUC system is 16.29 km/h and for SPSA 16.47 km/h. Figure 6.16 presents the trajectories of the mean speed for the first 3 replications when SPSA algorithm is applied to TUC strategy. The trajectories of the rest 7 simulation runs are excluded as they are all quite similar.

Table 6.2: Comparison of the average mean speed (MS) with and without the application of SPSA algorithm to TUC strategy (demand scenario 1).

	No AFT		SPSA		
ID	MS (km/h)	St.Dev.	MS (km/h)	St.Dev.	Improvement
1	16.41	0.68	16.08	0.64	-2.01%
2	16.53	0.80	16.52	0.58	-0.06%
3	16.09	0.51	16.25	0.94	0.99%
4	16.30	0.56	16.73	0.73	2.64%
5	16.40	0.66	16.57	0.80	1.04%
6	16.29	0.73	16.37	0.51	0.49%
7	16.38	0.73	16.41	0.56	0.18%
8	16.23	0.86	16.53	0.66	1.85%
9	16.07	0.68	16.66	0.73	3.67%
10	16.18	0.90	16.56	0.68	2.35%
Average	16.29	0.71	16.47	0.68	1.10%

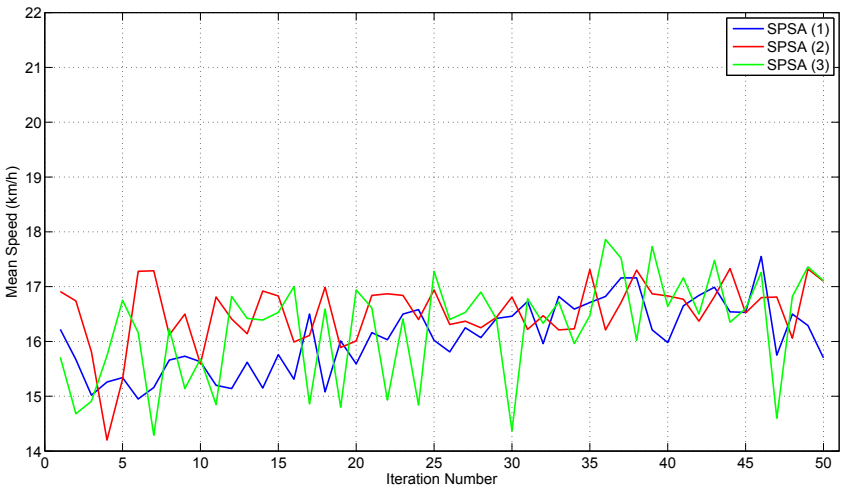

Figure 6.16: Mean speed for demand scenario 1 (TUC/SPSA–replications 1, 2 and 3).

6.7.2 Global results for demand scenario 2

This section presents the simulation results obtained for the demand scenario 2 (high demand). Figures 6.17–6.26 compare the network-wide mean speed of the original TUC system (blue line) versus TUC system combined with AFT algorithm (red line) delivered for this demand scenario. In all figures, it can be seen that the application of the AFT algorithm leads to better performance than the original TUC system itself. More precisely, AFT algorithm achieves to optimize the overall system performance within few days (iteration number in figures), by efficiently fine-tuning the design parameters for all TUC's control modules, while avoiding decreasing the daily mean speed lower than the initial point.

In all figures, the trajectory of the system performance (mean speed) is persistently increasing until it converges to a local maximum value. Note that the oscillations appearing in the blue lines of the figures (no AFT case), are due to the ±5% daily random perturbations applied to the demand scenario. The same perturbations are also applied to the AFT case; However, the oscillations of the red lines (after the convergence of the algorithm) are clearly lower. Finally, the figures depict slight differences between the replications due to the stochasticity of the simulator and the random demand perturbations. However, all the simulations experiments demonstrate the superiority of AFT algorithm over the manually fine-tuned TUC system.

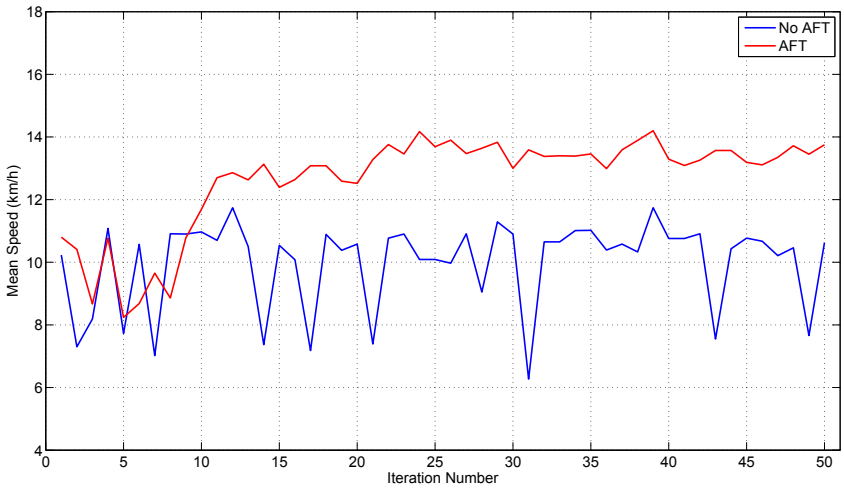

Figure 6.17: Mean speed for demand scenario 2 (TUC/AFT–replication 1).

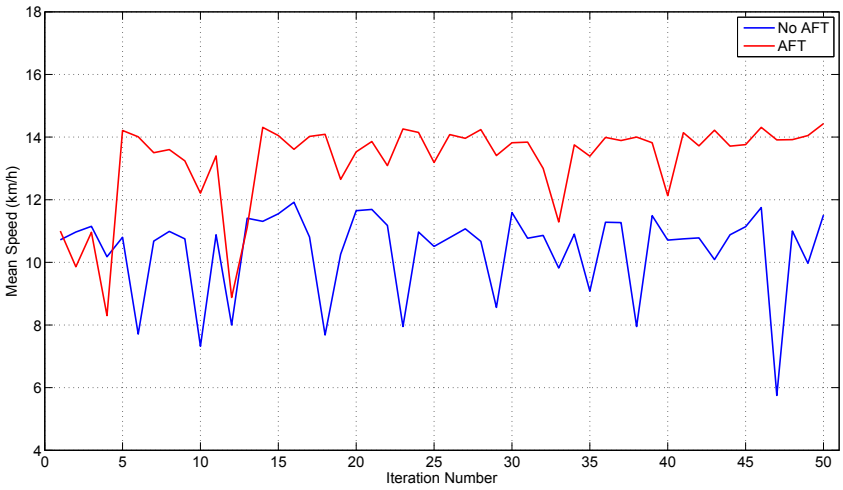

Figure 6.18: Mean speed for demand scenario 2 (TUC/AFT–replication 2).

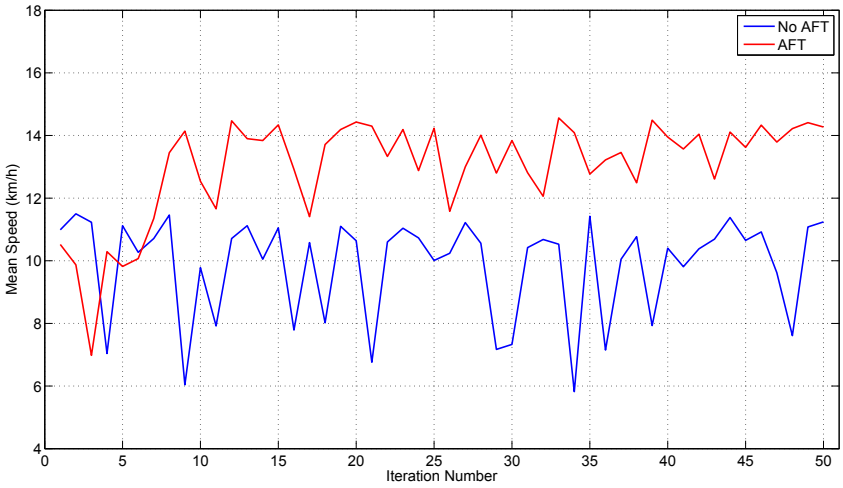

Figure 6.19: Mean speed for demand scenario 2 (TUC/AFT–replication 3).

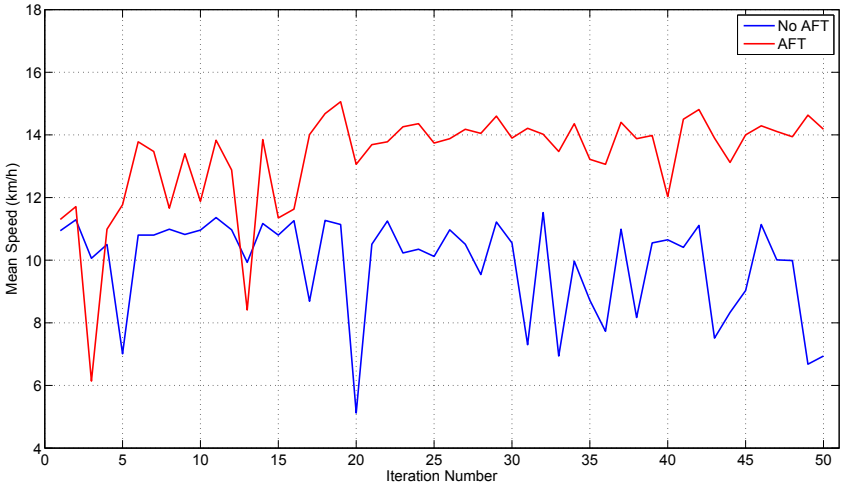

Figure 6.20: Mean speed for demand scenario 2 (TUC/AFT–replication 4).

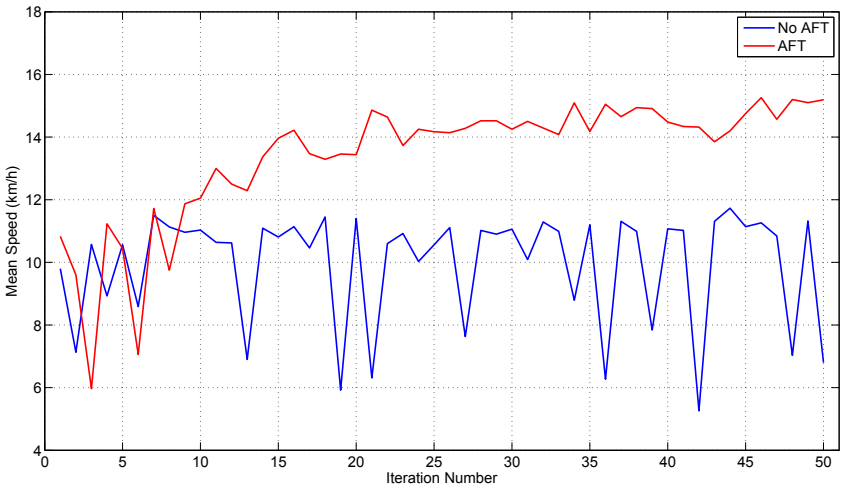

Figure 6.21: Mean speed for demand scenario 2 (TUC/AFT–replication 5).

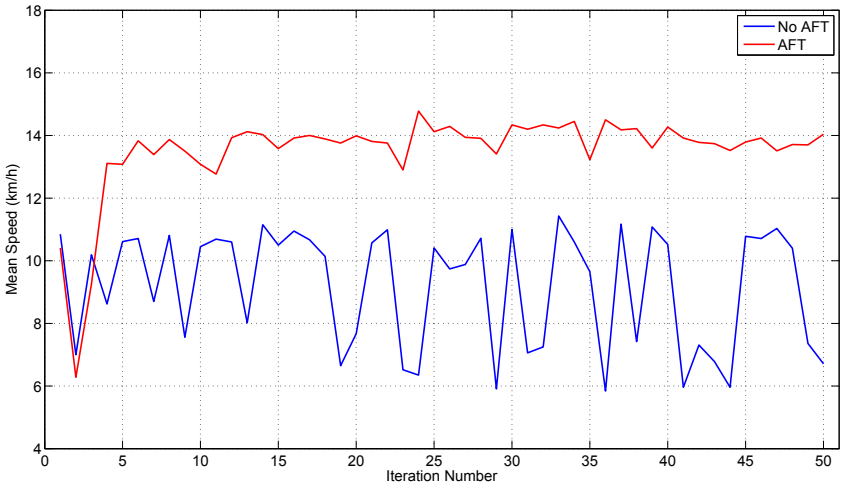

Figure 6.22: Mean speed for demand scenario 2 (TUC/AFT–replication 6).

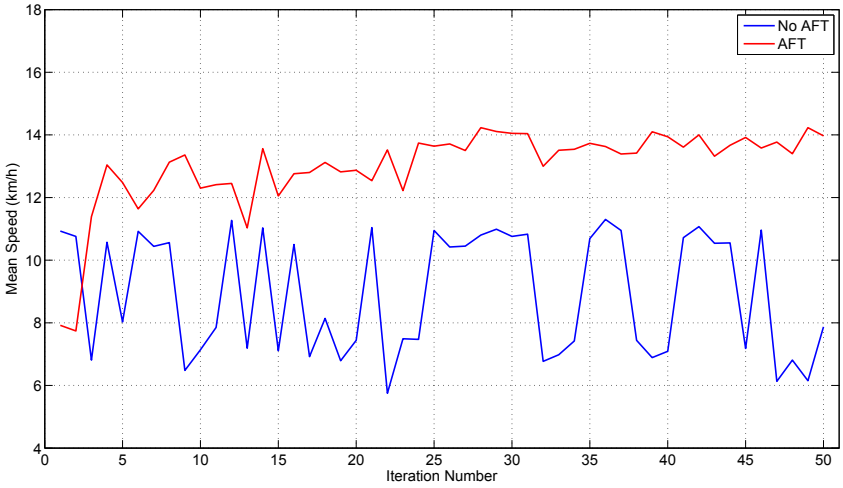

Figure 6.23: Mean speed for demand scenario 2 (TUC/AFT–replication 7).

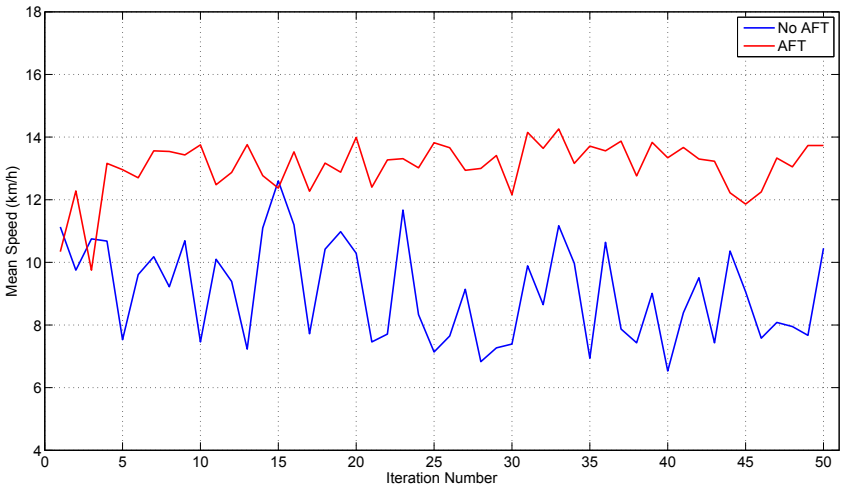

Figure 6.24: Mean speed for demand scenario 2 (TUC/AFT–replication 8).

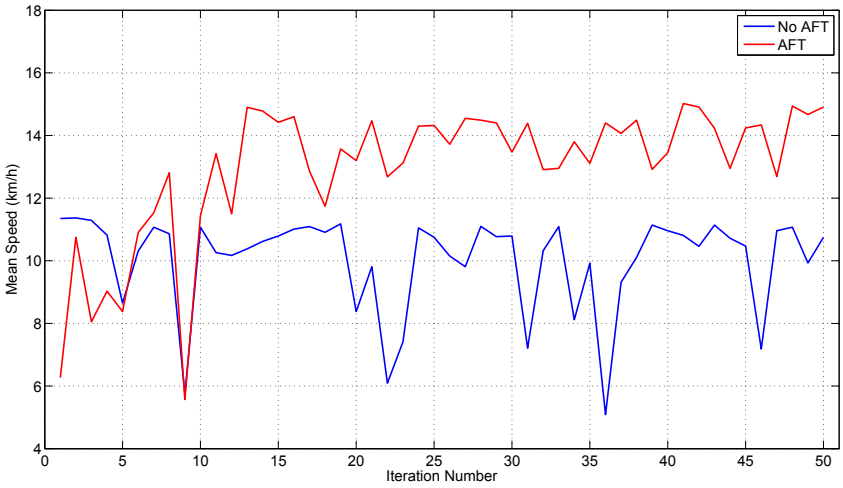

Figure 6.25: Mean speed for demand scenario 2 (TUC/AFT–replication 9).

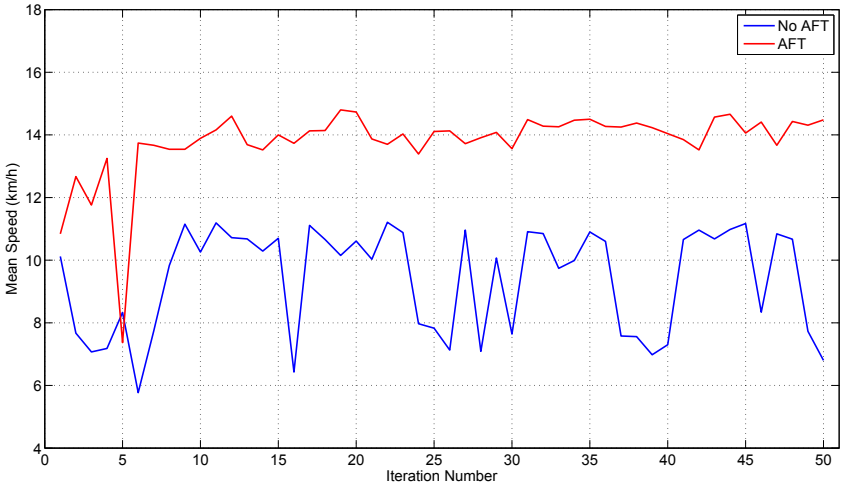

Figure 6.26: Mean speed for demand scenario 2 (TUC/AFT–replication 10).

Table 6.3 displays the mean speed for the original TUC system and the mean speed when using AFT for the system fine-tuning, for each replication. Also, the standard deviation of the mean speed due to the ±5% daily random perturbations of the demand is presented. Again, AFT algorithm is learning the traffic system dynamics during the first iterations (days) and then converges to a local optimal solution. Thus, the results for AFT are presented for both the whole simulation horizon (column 4 in Table 6.3) and after the convergence of AFT (column 7 in Table 6.3). Finally, Table 6.3 displays the number of days needed for AFT to converge as well as the percentage improvement over the no AFT case.

It can be seen that the use of AFT algorithm leads to an average improvement of the system performance of some 36% for this demand scenario, and, if the learning period is excluded (which is 7.9 days on average) the improvement increases to some 41%. The average mean speed for the original TUC system is 9.67 km/h and after the convergence of AFT this speed is increased to 13.61 km/h. Also, the average standard deviation of the daily mean speed is decreased to the half (from 1.63 km/h to 0.82 km/h) after the application of AFT, leading to smaller daily variations after the algorithm's convergence.

Table 6.3: Comparison of the average mean speed (MS) with and without the application of AFT algorithm to TUC strategy (demand scenario 2).

	No AFT		AFT (all 50 days)		Conv.	AFT (after convergence day)		
ID	MS	St.Dev.	MS	Improvement	Day	MS	St.Dev.	Improvement
1	9.99	1.41	12.63	26.40%	12	13.33	0.44	33.35%
2	10.39	1.36	13.24	27.41%	5	13.41	1.25	29.03%
3	9.87	1.60	13.02	31.94%	8	13.49	0.91	36.67%
4	9.90	1.56	13.27	34.07%	6	13.55	1.18	36.91%
5	9.97	1.79	13.40	34.42%	15	14.37	0.55	44.14%
6	9.19	1.88	13.52	47.06%	6	13.85	0.42	50.61%
7	8.95	1.92	13.00	45.33%	4	13.22	0.77	47.74%
8	9.06	1.56	13.06	44.13%	4	13.14	0.76	44.93%
9	10.04	1.54	12.97	29.27%	13	13.86	0.90	38.09%
10	9.39	1.67	13.79	46.77%	6	13.93	1.06	48.31%
Average	9.67	1.63	13.19	36.33%	7.9	13.61	0.82	40.70%

Once again, the application of SPSA algorithm to the same replications did not manage to provide any satisfactory results due to the scale and complexity of the problem. Table 6.4 displays the obtained results for all simulated replications. Although the trajectories seem to have an increasing tendency (there are actually 3 replications with some 5% improvement), SPSA fails to increase the mean speed of the network to a desirable level. The average mean speed for the original TUC system is 9.67 km/h and 9.88 km/h after the implementation of SPSA. Figure 6.27 presents the mean speed trajectories for the first 3 replications of SPSA algorithm application to TUC strategy. The trajectories of the rest 7 simulation runs are excluded as they are all quite similar.

Table 6.4: Comparison of the average mean speed (MS) with and without the application of SPSA algorithm to TUC strategy (demand scenario 2).

	No AFT		SPSA		
ID	MS (km/h)	St.Dev.	MS (km/h)	St.Dev.	Improvement
1	9.99	1.41	9.84	1.12	-1.50%
2	10.39	1.36	10.68	1.48	2.79%
3	9.87	1.60	9.93	1.66	0.61%
4	9.90	1.56	9.87	1.36	-0.30%
5	9.97	1.79	10.42	1.14	4.51%
6	9.19	1.88	9.42	1.46	2.50%
7	8.95	1.92	9.08	1.24	1.45%
8	9.06	1.56	9.16	1.77	1.10%
9	10.04	1.54	10.57	1.44	5.28%
10	9.39	1.67	9.77	1.26	4.05%
Average	9.67	1.63	9.88	1.39	2.17%

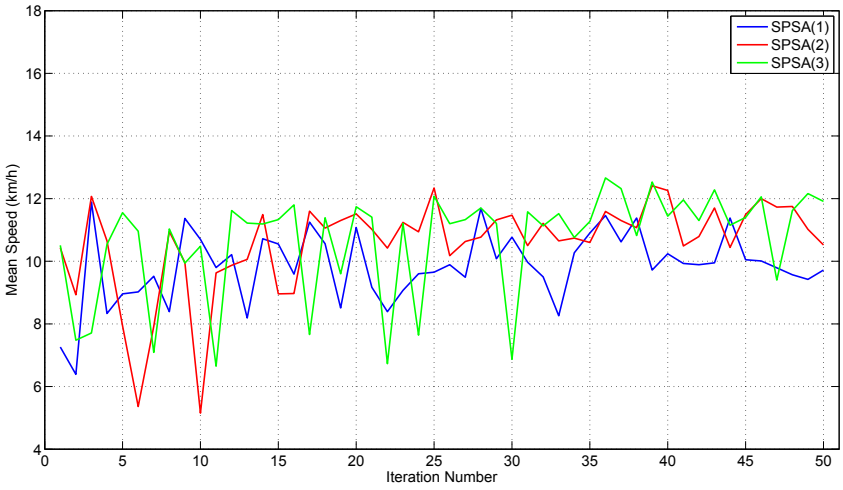

Figure 6.27: Mean speed for demand scenario 2 (TUC/SPSA–replications 1, 2 and 3).

6.7.3 Detailed results

Figures 6.28 and 6.29 display the importance factors of the network links according to the optimal solution of the AFT algorithm for scenarios 1 and 2, respectively. Green color indicates low importance ($b_i \leq 0.7$), black color indicates medium importance ($0.7 < b_i < 1.3$), and red color indicates high link importance ($b_i \geq 1.3$). It can be seen that AFT algorithm increases the importance factors for network links along the main entrance to and exit from the city centre (junctions 16, 15, 14, 13, 12, 5, 4), while for others – not so crucial for the overall system performance – the corresponding weights are decreased. Although there are links that have the same color for both scenarios, there are others that in scenario 1 are green and in scenario 2 red and vice versa. An explanation for this, is that AFT converges to a local optimal solution which fine-tunes the importance factors and optimizes the traffic control system performance, but which also depends on the special characteristics of each demand scenario.

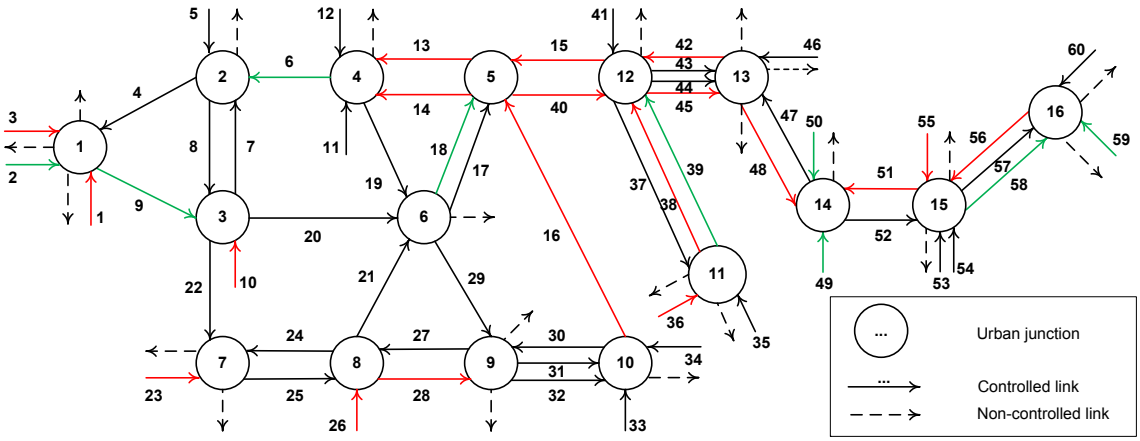

Figure 6.28: Network links importance (green for low, black for medium, red for high) to the split module of TUC according to AFT algorithm for scenario 1.

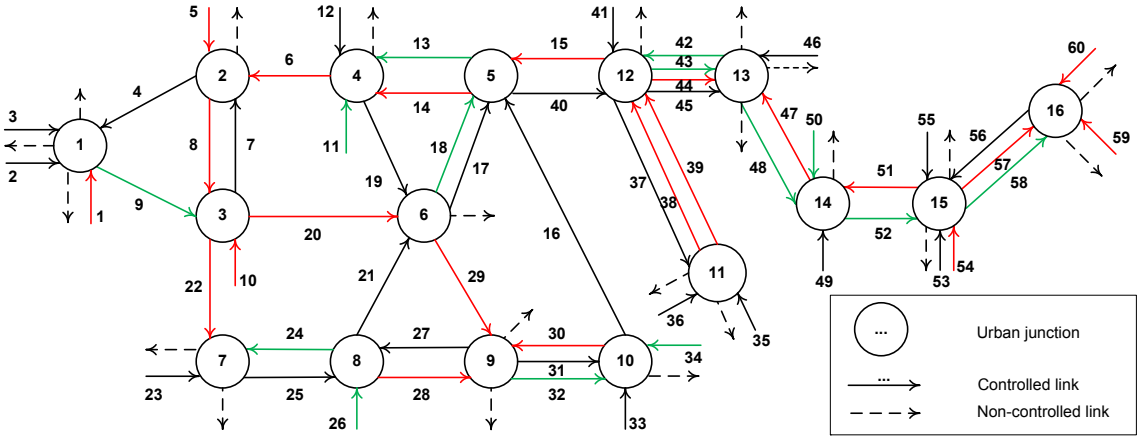

Figure 6.29: Network links importance (green for low, black for medium, red for high) to the split module of TUC according to AFT algorithm for scenario 2.

In the sequel, we report on some selected results focussing on the city's main shopping district (junction 5 in Figure 6.5).

Regarding the split control module of TUC strategy, Figure 6.30 compares the time evolution of the design parameters b_{15} and b_{18} under the use of the AFT algorithm, with some optimized values for these parameters. The optimized values come from a manual fine-tuning procedure, performed in the past by human experts in a field evaluation of TUC strategy in Chania network [37]. The manual tuning of b_{15} and b_{18} led to the optimized weights of 1.8 and 0.6 respectively. This is a quite reasonable link weighting, as link 15 is a crucial link in the main arterial of the city centre, contrary to link 18 which does not carry

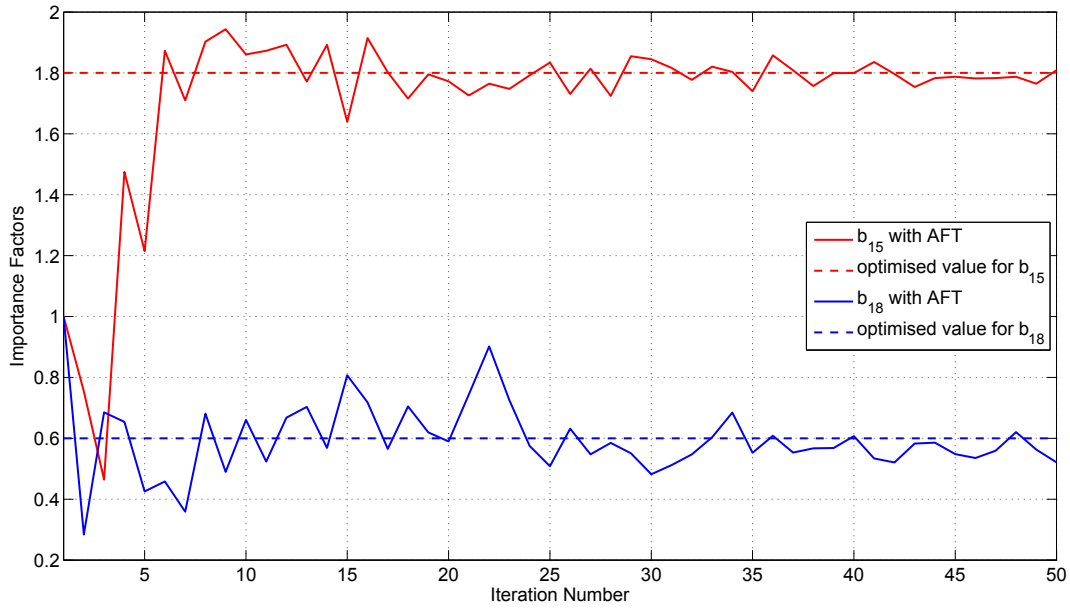

Figure 6.30: Trajectories of the split parameters b_{15}, b_{18} and corresponding optimized values (scenario 1 – replication 7).

substantial traffic loads. AFT algorithm starts from the initial weights $b_{15} = b_{18} = 1$ and by iteratively tuning their values converges to weights that are seen to be close to the roughly optimized values.

Figure 6.31 displays the aforementioned design parameters for demand scenario 2. The weights again converge close to the optimized values, although they are slightly different than in scenario 1 due to different traffic conditions. What is clear from both figures is that link 15 is more important for the network mean speed than link 18. Note that this holds for many other network links not shown here (see Figures 6.28–6.29 for a general view). The results of Figures 6.30–6.31 demonstrate clearly that the proposed algorithm is a feasible and viable solution for automated parameter fine-tuning of such systems.

Continuing, we illustrate the impact of AFT algorithm to the cycle control module of TUC strategy. The tuning of the design parameters $\theta_2 = (K_1, K_2, \sigma_{N_1}, \sigma_{N_2}, \sigma_{cr})$ by AFT changes the reaction of the cycle control feedback-regulator (6.6) and achieves to notably improve the mean speed of the simulated network. Figure 6.32 displays the initial cycle control regulator and the fine-tuned cycle control regulators after the convergence of AFT algorithm for scenarios 1 and 2, respectively and for different values of the network load $\sigma(t_c)$. Recall, that $\sigma(t_c) \in [0, 1]$ is the average space occupancy for some pre-specified percentage of network links over the last cycle control period. Figure 6.32 depicts the three different regulators for $\sigma(t_c) \in [0, 0.7]$.

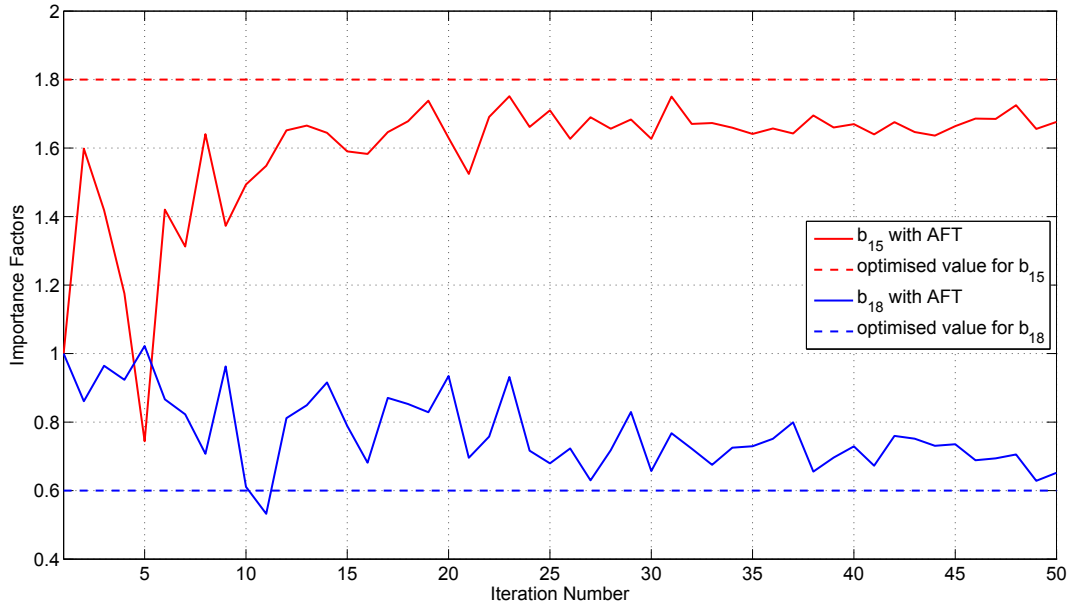

Figure 6.31: Trajectories of the split parameters b_{15}, b_{18} and corresponding optimized values (scenario 2 – replication 3).

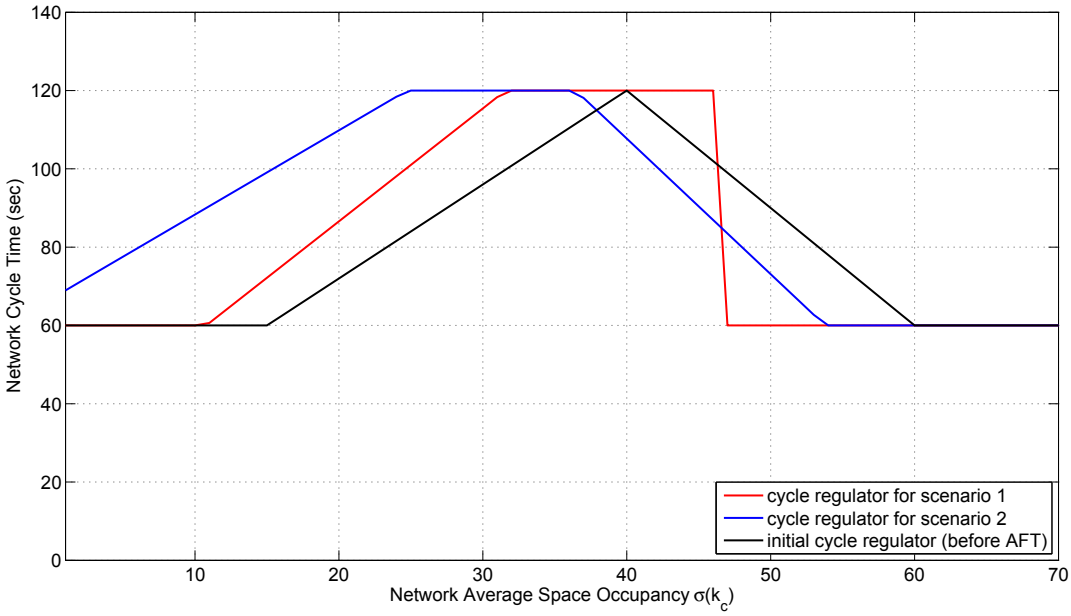

Figure 6.32: Initial cycle regulator and the cycle regulators after the convergence of AFT for different values of average network loads (the regulators correspond to replications 7, 3 for scenarios 1, 2 respectively).

The initial cycle control regulator (see equation (6.6)) consists of one monotonically increasing function for undersaturated traffic conditions ($\sigma_{cr} < 0.4$) and one monotonically decreasing function for saturated traffic conditions ($\sigma_{cr} > 0.4$). The maximum cycle $C_{\max} = 120s$ is applied for the critical occupancy $\sigma_{cr} = 0.4$. After the convergence of AFT algorithm, a cycle regulator with three regimes is obtained for both simulated scenarios, that is, one regime with increasing cycle periods for undersaturated traffic conditions, one regime with maximum cycle periods $C_{\max} = 120s$ (for $0.32 \leq \sigma_{cr} \leq 0.46$ and $0.25 \leq \sigma_{cr} \leq 0.36$ for scenarios 1 and 2, respectively) for traffic conditions that require to increase the capacity of the network, and one regime with decreasing cycle periods for saturated traffic conditions (see Figure 6.32). For the third regime the network faces severe congestion problems due to queue spillovers and partial gridlocks that lead to a strong performance deterioration. Again, there are slight differences between the fine-tuned cycle control regulators of scenario 1 and 2, which depend on the different traffic characteristics.

In general, the derived trapezoidal shape (see Figure 6.32) of the cycle control regulators, over the saturation level of the network, outperforms the initial cycle regulator as it is shown by the overall system performance. Eventually, three traffic regimes may be identified: (a) undersaturated traffic conditions, (b) critical traffic conditions, and (c) saturated traffic conditions. In that way, the cycle control module of TUC applies appropriate cycle times for each case, that is, smaller cycle times for regimes (a) and (c) and maximum cycle time for regime (b) so as to maximize the network's capacity. Note that the starting points of regimes (b) and (c) occur in different values of σ_{cr} for demand scenarios 1 and 2, which means that the real-time implementation of AFT algorithm could be vital for the overall system performance.

In order to elaborate on the simulation results, Figure 6.33 presents another simulation run for demand scenario 2. The blue line represents the actual mean speed of the network (as provided by the simulator). The red line represents the estimation/prediction that AFT algorithm does (in the previous iteration) about the mean speed. Based on the approximator $\widehat{J}(\theta, x)$, AFT picks the best of the random perturbation candidates to be applied for the next day (according to the estimated performance index). The red line depicts this estimation about every algorithm iteration and the blue line the actual performance after the set of parameters has been applied.

It can be seen, that the prediction of AFT for the next day mean speed is quite accurate. The values of $\widehat{J}(\theta(k+1), \bar{x}(k+1))$ and $J(\theta(k+1), x(k+1))$ are not equal but the drift of the red line follows the drift of the average value of the blue line. This is enough to provide the algorithm with efficient behavior as the estimation of the exact actual value of the mean speed is not a prerequisite.

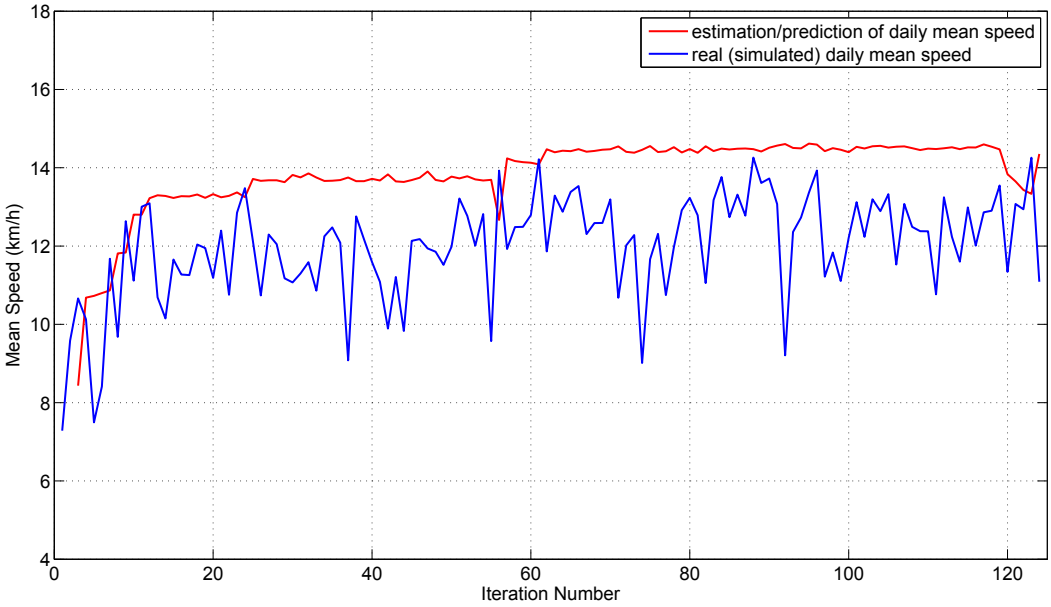

Figure 6.33: Simulation model of the Chania urban road network.

Finally, Figure 6.34 provides a representation of the input-output mapping of the approximator $\widehat{J}(\theta, x)$ during the iteration number 25 for the same simulation. In order to evaluate the candidate random perturbations AFT is based on the model that derives by the fitting of previous data to $\widehat{J}(\theta, x)$. In most algorithm iterations, this fitting provides an unbiased approximation or has a very small bias practically equal to 0 (Figure 6.34).

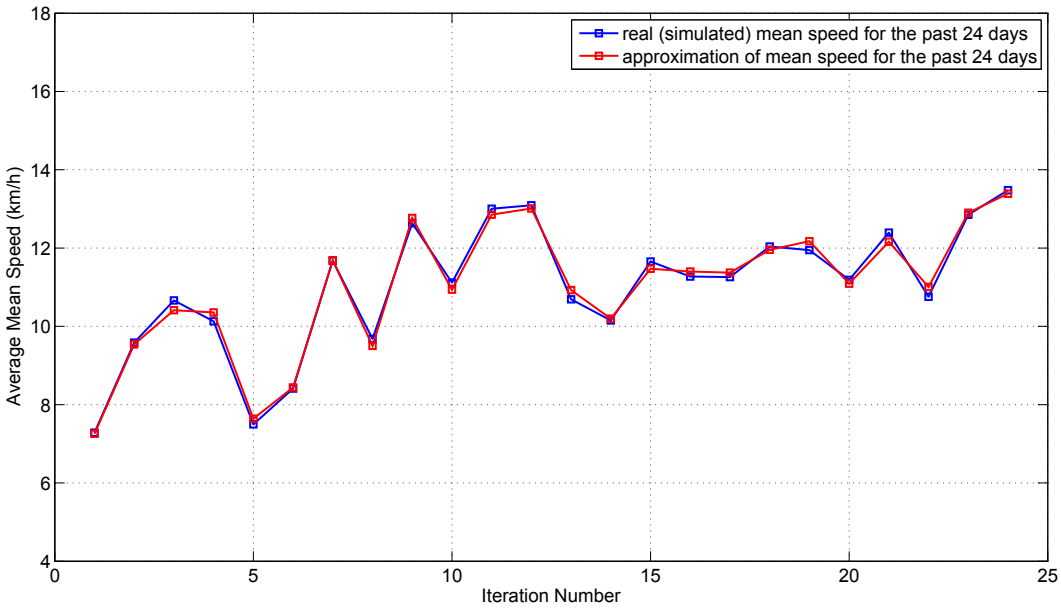

Figure 6.34: Simulation model of the Chania urban road network.

6.8 Concluding remarks

This chapter investigated the efficiency of AFT algorithm to the problem of optimizing the design parameters of a large-scale traffic control system composed by distinct and mutually-interacting modules. This adaptive optimization methodology aims at replacing the conventional manually-based optimization with a fully-automated procedure. Extensive simulation experiments have been conducted for the signal control problem of the large-scale traffic network of the city of Chania, where the design parameters of three distinct and mutually-interacting modules of TUC strategy were fine-tuned by AFT algorithm. The simulation results, as well as the comparison to the base-case, where the aforementioned design parameters of TUC system were manually fine-tuned to virtual perfection by the system operators, demonstrate the algorithm's efficiency and feasibility.

The design parameters of the split, cycle and offset control modules of TUC have been considered for fine-tuning. It was demonstrated that the application of AFT algorithm to the signal control strategy TUC leads to better network performance (in terms of daily mean speed) compared to the original TUC system. This, underlines the superiority of the fully-automated optimization procedure, pursued by the AFT algorithm, even in the case that the

design parameters are already manually fine-tuned by field experts.

Regarding the design parameters of split control module, it was shown that AFT algorithm increases the weight of the importance factors for network links along the main entrance to and exit from the city centre, while for others, not so crucial, the corresponding weights are decreased. Furthermore, it was shown that AFT algorithm converges to sets of quite reasonable design parameters, close to the roughly optimized values provided by the system operators for the original TUC system. Finally, regarding the design parameters of the cycle control module, it was shown that AFT algorithm leads to cycle control regulators with a trapezoidal shape, with three corresponding traffic regimes, which outperform the initial cycle control regulator.

The average computational time for every iteration of AFT algorithm is again a few seconds, which means that the implementation of the algorithm in a real-time large-scale application would be feasible, regardless the type of the operating Traffic Control System. Finally, it should be stressed that AFT algorithm could be also utilized as an off-line network optimization tool, for calculating optimum sets of design parameters for large-scale traffic control systems of any type, since its system dynamics and controls (equations (4.1), (4.2)) and related performance criterion (equation (4.4)) incorporate all necessary network characteristics.

Chapter 7

Conclusions and future work

This chapter concludes the thesis, summarizing its findings and results. Section 7.1 highlights the final remarks and Section 7.2 presents some future perspectives which can help to the extension of the achieved results.

7.1 Thesis concluding remarks

Despite the continuous advances in the fields of control and computing, the design and deployment of efficient LNTCSs is often based on simplified models for the system dynamics. This is mainly a consequence of the complexity and the nonlinearity of the practical traffic control system applications. As a result, the majority of LNTCSs require a tedious fine-tuning of their design parameters prior and during the actual system operation. Currently, the fine-tuning process is performed by experienced personnel, based on field observations via experimentation with different combinations of design parameters, without the use of a systematic approach.

This thesis introduced and analyzed a new learning/adaptive algorithm that can provide with convergent, efficient and safe automatic adaptive fine-tuning of general LNTCS. The proposed methodology combines the principles of Adaptive Optimization and Stochastic Approximation methods. The LNTCS fine-tuning problem is formulated as an optimization problem where the objective function cannot be computed directly (i.e. its analytical form is not known) but only estimated via observations. An approximator (i.e. neural network or neuro-fuzzy dynamical system) is used in order to estimate the value of the unknown objective function, which corresponds to the overall (measurable) system performance. The algorithm runs iteratively and updates the control parameters vector (according to the objective function approximation and the adaptive optimization scheme), so as to achieve better system performance. A detailed analysis of the algorithm as well as a step-by-step application

description was provided.

The efficiency and online feasibility of AFT algorithm was investigated through extensive simulation experiments for two LNTCS. The first test case is a large-scale ramp metering control problem. A multivariable ramp metering regulator is applied to the stretch of the Monash motorway in Melbourne, Australia. The latter test case corresponds to the application of an urban signal control strategy to the road network of Chania, in Greece . In both simulated cases, the simulation results illustrate that AFT (almost sure) converges to a (local) maximum of the performance index. This indicates that AFT is an efficient and feasible algorithm for real-time fine-tuning of the design parameters of general LNTCS.

7.2 Future perspectives

The results of this thesis are quite promising. Two future directions which can help to the extension of the achieved results are:

- Online field application of the presented algorithm. The performance of AFT could be evaluated through a real-time LNTCS field implementation.
- Application to other large-scale control problems different than traffic systems.

AFT algorithm is a general straightforward fine-tuning methodology that is not limited in LNTCS. It could also be applied to other large-scale control systems that call for fine-tuning, independently of the physical process under control. Of course, every application domain has its own special characteristics, so a good knowledge of the considered domain is required for a successful implementation. Nevertheless, AFT can be applied for automated calibration to control systems of any structure or domain, e.g. chemical, mechanical, robotics, etc. Further application results and possible field implementation could demonstrate the algorithm's feasibility and justify its theoretical convergence properties.

Bibliography

[1] *Aimsun Users Manual Version 6*, TSS-Transport Simulation Systems, Barcelona, Spain, 2008.

[2] K. Aboudolas, M. Papageorgiou, A. Kouvelas, and E. Kosmatopoulos, *A Rolling-Horizon Quadratic-Programming Approach to the Signal Control Problem in Large-Scale Congested Urban Road Networks*, Transportation Research – Part C, 18 (2010), pp. 680–694.

[3] G. Abu-Lebdeh and R. F. Benekohal, *Development of traffic control and queue management procedures for oversaturated arterials*, Transportation Research Record, (1997), pp. 119–127.

[4] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, John Wiley & Sons, Inc., New York, 2nd ed., 1993.

[5] D. P. Bertsekas and J. N. Tsitsiklis, *Neuro-Dynamic Programming*, Athena Scientific, Belmont, Massachusetts, 1996.

[6] J. R. Blum, *Multidimensional stochastic approximation methods*, Annals of Mathematical Statistics, 25 (1954b), pp. 737–744.

[7] V. S. Borkar, *Stochastic Approximation: A Dynamical Systems Viewpoint*, Cambridge University Press, New York, 2008.

[8] R. Burnett, *Application of stochastic optimization to collision avoidance*, in Proceedings of the 2004 American Control Conference, vol. 3, 2004, pp. 2789–2794.

[9] M. C. Choy, D. Srinivasan, and R. Cheu, *Neural networks for continuous online learning and control*, IEEE Transactions on Neural Networks, 17 (2006), pp. 1511–1531.

[10] C. Diakaki, *Integrated Control of Traffic Flow in Corridor Road Networks*, PhD thesis, Technical University of Crete, Chania, Greece, 1999.

[11] C. Diakaki, V. Dinopoulou, K. Aboudolas, and M. Papageorgiou, *Final system development report*, Deliverable 7, Project SMART NETS, Information Society Technologies Office, Brussels, Belgium, 2002.

[12] C. Diakaki, V. Dinopoulou, K. Aboudolas, M. Papageorgiou, E. Ben-Shabat, E. Seider, and A. Leibov, *Extensions and new applications of the Traffic-responsive Urban Control strategy: Coordinated signal control for urban networks*, Transportation Research Record, (2003), pp. 202–211.

[13] C. Diakaki, M. Papageorgiou, and K. Aboudolas, *A Multivariable Regulator Approach to Traffic-Responsive Network-Wide Signal Control*, Control Engineering Practice, 10 (2002), pp. 183–195.

[14] V. Dinopoulou, E. Kosmatopoulos, and M. Papageorgiou, *Application of the traffic signal control strategy tuc to networks with public transport priority*, in Proceedings International Conference on Modern Trends in Traffic Signalling and Telematic Systems, Patra, Greece, 2004.

[15] Y. Ermoliev, *On the method of generalized stochastic gradients and quasi-fejer sequences*, Cybernetics, 5 (1969), pp. 208–220.

[16] V. Fabian, *Stochastic approximation of minima with improved asymptotic speed*, Annals of Mathematical Statistics, 38 (1967), pp. 191–200.

[17] V. Fabian, *Stochastic approximation*, Optimizing Methods in Statistics, (J. S. Rustigi, ed.), Academic Press, New York, (1971), pp. 439–470.

[18] J. L. Farges, J. J. Henry, and J. Tufal, *The PRODYN real-time traffic algorithm*, in Proceedings 4th IFAC Symposium on Trasportation Systems, Baden-Baden, Germany, 1983, pp. 307–312.

[19] M. C. Fu and S. D. Hill, *Optimization of discrete event systems via simultaneous perturbation stochastic approximation*, IIE Transactions, 29 (1997), pp. 223–243.

[20] M. C. Fu and J. Q. Hu, *Conditional Monte Carlo: Gradient Estimation and Optimization Applications*, Kluwer Academic, Boston, 1997.

[21] N. H. Gartner, *OPAC: A demand-responsive strategy for traffic signal control*, Transportation Research Record, (1983), pp. 75–84.

[22] D. C. Gazis, *Traffic control, time-space diagrams and networks*, Traffic Control-Theory and Instrumentation, (T. R. Horton, ed.), Plenum, New York (1965), pp. 47–63.

[23] D. C. Gazis and R. B. Potts, *The Oversaturated Intersection*, in Proceedings 2nd International Symposium on Traffic Theory, London, UK, 1963, pp. 221–237.

[24] A. E. Gelfand and A. F. M. Smith, *Sampling-based approaches to calculating marginal densities*, Journal of the American Statistical Association, 85 (1990), pp. 399–409.

[25] N. B. Goldstein and K. S. P. Kumar, *A decentralized control strategy for freeway regulation*, Transportation Research – Part B, 16 (1982), pp. 279–290.

[26] H. Hadj-Salem and M. Papageorgiou, *Ramp metering impact on urban corridor traffic: Field results*, Transportation Research – Part A, 29 (1995), pp. 303–319.

[27] S. Hill and M. Fu, *Transfer optimization via simultaneous perturbation stochastic approximation*, in Proceedings of the 27th Winter Simulation Conference, 1995, pp. 242–249.

[28] G.-B. Huang, L. Chen, and C.-K. Siew, *Universal approximation using incremental constructive feedforward networks with random hidden nodes*, IEEE Transactions on Neural Networks, 17 (2006), pp. 879–892.

[29] P. B. Hunt, D. I. Robertson, R. D. Bretherton, and M. C. Royle, *The SCOOT on-line traffic signal optimization technique*, Traffic Engineering and Control, 23 (1982), pp. 190–192.

[30] J.-S. R. Jang, *Anfis: Adaptive-network-based fuzzy inference systems*, IEEE Transactions on Systems, Man, and Cybernetics, 23 (1993), pp. 665–685.

[31] H. Jula, M. Dessouky, P. Ioannou, and A. Chassiakos, *Container movement by trucks in metropolitan networks: modeling and optimization*, Transportation Research – Part E, 41 (2005), pp. 235–259.

[32] J. Kiefer and J. Wolfowitz, *Stochastic estimation of a regression function*, Annals of Mathematical Statistics, 23 (1952), pp. 462–466.

[33] N. Kleinman, S. Hill, and V. Ilenda, *SPSA/SIMMOD optimization of air traffic delay cost*, in Proceedings of the 1997 American Control Conference, vol. 2, 1997, pp. 1121–1125.

[34] M. Koch, D. Chin, and R. Smith, *Network-wide approach to optimal signal light timing for integrated transit vehicle and traffic operations*, in Proceedings of the 7th National Conference on Light Rail Transit, vol. 2, 1997, pp. 126–131.

[35] J. Koronacki, *Random-seeking methods for the stochastic unconstrained optimization*, International Journal of Control, 21 (1975), pp. 517–527.

[36] E. Kosmatopoulos, M. Christodoulou, and P. Ioannou, *Dynamical neural networks that ensure exponential identification error convergence*, Neural Networks, 10 (1997), pp. 299–314.

[37] E. Kosmatopoulos, M. Papageorgiou, C. Bielefeldt, V. Dinopoulou, R. Morris, J. Mueck, A. Richards, and F. Weichenmeier, *International Comparative Field Evaluation of a Traffic-Responsive Signal Control Strategy in Three Cities*, Transportation Research – Part A, 40 (2006), pp. 399–413.

[38] E. Kosmatopoulos, M. Polycarpou, M. Christodoulou, and P. Ioannou, *High-order neural network structures for identification of dynamical systems*, IEEE Transactions on Neural Networks, 6 (1995), pp. 422–431.

[39] E. B. Kosmatopoulos, *Adaptive control design based on adaptive optimization principles*, IEEE Transactions on Automatic Control, 53 (2008), pp. 2680–2685.

[40] E. B. Kosmatopoulos, *An adaptive optimization scheme with satisfactory transient performance*, Automatica, 45 (2009), pp. 716–723.

[41] E. B. Kosmatopoulos and A. Kouvelas, *Large-scale nonlinear control system fine-tuning through learning*, IEEE Transactions on Neural Networks, 20 (2009), pp. 1009–1023.

[42] E. B. Kosmatopoulos, M. Papageorgiou, A. Vakouli, and A. Kouvelas, *Adaptive fine-tuning of non-linear control systems with application to the urban traffic control strategy TUC*, IEEE Transactions on Control Systems Technology, 15 (2007), pp. 991–1002.

[43] A. Kouvelas, K. Aboudolas, M. Papageorgiou, and E. Kosmatopoulos, *A hybrid strategy for real-time traffic signal control of urban road networks*, in Proceedings 89th TRB Annual Meeting, Washigton, D.C., U.S.A., 2010.

[44] C.-M. Kuan and K. Hornik, *Convergence of learning algorithms with constant learning rates*, IEEE Transactions on Neural Networks, 2 (1991), pp. 484–489.

[45] L. Li, N. Tang, X. Mu, and F. Shi, *Implementation of traffic lights control based on petri nets*, IEEE Transactions on Intelligent Transportation Systems, 3 (2003), pp. 1749–1752.

[46] P.-c. B. Liu, M. Hansen, and A. Mukherjee, *Scenario-based air traffic flow management: From theory to practice*, Transportation Research – Part B, 42 (2008), pp. 685–702.

[47] D. P. Looze, P. K. Houpt, N. R. Sandell, and M. Athans, *On decentralized estimation and control with application to freeway ramp metering*, IEEE Transactions on Automatic Control, 23 (1978), pp. 268–275.

[48] P. R. Lowrie, *SCATS: The Sydney co-ordinated adaptive traffic system—Principles, methodology, algorithms*, in Proceedings IEE International Conference on Road Traffic Signalling, London, England, 1982, pp. 67–70.

[49] V. Maiorov and R. Meir, *Approximation bounds for smooth functions in $C(\Re^d)$ by neural and mixture networks*, IEEE Transactions on Neural Networks, 9 (1998), pp. 969–978.

[50] E. H. Mamdani and S. Assilian, *An experiment in linguistic synthesis with a fuzzy logic controller*, International Journal of Man-Machine Study, 7 (1975), pp. 1–13.

[51] D. P. Masher, D. W. Ross, P. J. Wong, P. L. Tuan, H. M. Zeidler, and S. Peracek, *Guidelines for design and operating of ramp control systems*, sri project 3340, Stanford Res. Inst. Rep. NCHRP 3–22, Menid Park, CA, 1975.

[52] K. McCabe and A. Riley, *A flexible approach to motorway control*, Atkins Technical Journal, 3, pp. 37–42.

[53] A. Messmer and M. Papageorgiou, *Metanet: A macroscopic simulation program for motorway networks*, Traffic Engineering and Control, 31 (1990), pp. 466–470.

[54] F. Middelham, H. Taale, J. Dibbits, and W. Fransen, *The assessment of the SCOOT system in Nijmegen*, IEE Conference Publications, 1996 (1996), pp. 66–70.

[55] P. Mirchandani and L. Head, *RHODES—A real–time traffic signal control system: Architecture, algorithms, and analysis*, in TRISTAN III (Triennial Symposium on Transportation Analysis), vol. 2, San Juan, Puerto Rico, 1998.

[56] M. Papageorgiou, *Multilayer control system design applied to freeway traffic*, IEEE Transactions on Automatic Control, 29 (1984), pp. 482–490.

[57] M. Papageorgiou, J. M. Blosseville, and H. Hadj-Salem, *Modeling and real-time control of traffic flow on the southern part of boulevard priphrique in Paris—Part II:*

Coordinated on-ramp metering, Transportation Research – Part A, 24 (1990), pp. 361–370.

[58] M. Papageorgiou, C. Diakaki, V. Dinopoulou, A. Kotsialos, and Y. Wang, *Review of Road Traffic Control Strategies*, Proceedings of the IEEE, 91 (2003), pp. 2043–2067.

[59] M. Papageorgiou, H. Hadj-Salem, and J. M. Blosseville, *Alinea: A local feedback control law for on-ramp metering*, U.S. Dept. Transp., Washington, DC, Transp. Res. Record 1320, (1991).

[60] M. Papageorgiou, H. Hadj-Salem, and F. Middelham, *Alinea local ramp metering: Summary of field results*, U.S. Dept. Transp., Washington, DC, Transp. Res. Record 1603, (1998).

[61] I. Papamichail, M. Papageorgiou, and E. Kosmatopoulos, *Modelling, configuration and simulation testing for the VicRoads ramp metering pilot project*, Report for VicRoads, Melbourne, Australia, Dynamic Systems and Simulation Laboratory, Greece, 2007.

[62] K. M. Passino, *Biomimicry for optimization, control, and automation*, Springer, London, 2005.

[63] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*, John Wiley, Princeton, N.J., 2007.

[64] M. Riedmiller and H. Braun, *RPROP – a fast adaptive learning algorithm*, Proceedings of ISCIS VII), Universitat, 1992.

[65] H. Robbins and S. Monro, *A stochastic approximation method*, Annals of Mathematical Statistics, 22 (1951), pp. 400–407.

[66] P. Sadegh, *Constrained optimization via stochastic approximation with a simultaneous perturbation gradient approximation*, Automatica, 33 (1997), pp. 889–892.

[67] J. Sanchez, M. Galan, and E. Rubio, *Applying a traffic lights evolutionary optimization technique to a real case: "Las Ramblas" area in Santa Cruz de Tenerife*, IEEE Transactions on Evolutionary Computation, 12 (2008), pp. 25–40.

[68] J. Shao, *Linear model selection by cross-validation*, Journal of the American Statistical Association, 88 (1993), pp. 486–494.

[69] J. Spall, *Multivariate stochastic approximation using a simultaneous perturbation gradient approximation*, IEEE Transactions on Automatic Control, 37 (1992), pp. 332–341.

[70] J. Spall and D. Chin, *Traffic-responsive signal timing for system-wide traffic control*, Transportation Research – Part C, 5 (1997), pp. 153–163.

[71] J. C. Spall, *A one-measurement form of simultaneous perturbation stochastic approximation*, Automatica, 33 (1997), pp. 109–112.

[72] J. C. Spall, *Introduction to Stochastic Search and Optimization: Estimation, Simulation, and Control*, John Wiley, Hoboken, N.J., 2003.

[73] M. Stone, *Cross-validity choice and assessment of statistical predictors*, Journal of the Royal Statistical Society, Series B, 39 (1974), pp. 111–174.

[74] M. Sugeno and K. Tanaka, *Successive identification of a fuzzy model and its application to prediction of complex systems*, Fuzzy Sets and Systems, 42 (1991), pp. 315–334.

[75] T. Takagi and M. Sugeno, *Fuzzy identification of systems and its applications to modeling and control*, IEEE Transactions on Systems, Man, and Cybernetics, 15 (1985), pp. 116–132.

[76] H. M. Zhang, S. G. Ritchie, and R. Jayakrishnam, *Coordinated traffic-responsive ramp control via nonlinear state feedback*, Transportation Research – Part C, 9 (2001), pp. 337–352.