An Autonomic Methodology for Embedding Self-tuning Competence in Online Traffic Control Systems

Anastasios Kouvelas, Diamantis Manolis, Elias Kosmatopoulos, Ioannis Papamichail, and Markos Papageorgiou

Abstract Recent advances in technology, control and computer science play a key role towards the design and deployment of the next generation of intelligent transportation systems (ITS). The architecture of such complex systems is crucial to include supporting algorithms that can embody autonomic properties within the existing ITS strategies. This chapter presents a recently developed adaptive optimization algorithm that combines methodologies from the fields of traffic engineering, automatic control, optimization and machine learning in order to embed self-tuning properties in traffic control systems. The derived adaptive fine-tuning (AFT) algorithm comprises an autonomic tool that can be used in online ITS applications of various types, in order to optimize their performance by automatically fine-tuning the system's design parameters. The algorithm has been evaluated in simulation experiments, examining its ability and efficiency to fine-tune in real time the design parameters of a number of traffic control systems, including signal control for urban road networks. Field results are in progress for the urban network of Chania, Greece, as well as for energy-efficient building control. Some promising preliminary field results for the traffic control problem of Chania are presented here.

Keywords Intelligent transportation systems • Self-tuning • Adaptive fine-tuning

A. Kouvelas
EPFL ENAC IIC LUTS, CH-1015 Lausanne, Switzerland
e-mail: tasos.kouvelas@epfl.ch

D. Manolis • I. Papamichail (✉) • M. Papageorgiou
Dynamic Systems and Simulation Laboratory, Technical University of Crete, 73100 Chania, Greece
e-mail: dmanolis@dssl.tuc.gr; ipapa@dssl.tuc.gr; markos@dssl.tuc.gr

E. Kosmatopoulos
Democritus University of Thrace, 67100 Xanthi, Greece
e-mail: kosmatop@ee.duth.gr

T.L. McCluskey et al. (eds.), *Autonomic Road Transport Support Systems*, Autonomic Systems, DOI 10.1007/978-3-319-25808-9_13

1 Introduction

Despite the continuous advances in the fields of control and computing, the design and deployment of efficient large-scale traffic control systems remains a significant objective. This is mainly due to the complexity and the strong nonlinearities involved in the modelling of traffic flow processes. Practical control design approaches are often based on simplified models for the system dynamics, as the use of more complex models is virtually infeasible in most real-life applications. As a result, although the derived regulators are theoretically optimal, they usually exhibit suboptimal performance.

As the complexity of systems grows, the need to build into them the means to manage and maintain themselves becomes necessary, particularly in the case of large-scale, heterogeneous control systems. Systems need to be self-directing, self-configuring, self-maintaining, self-protecting and self-optimizing. One consequence of self-managing systems is that their interaction with people is set more at a "service" level than a "command" level. As a result, a traffic control centre manager will interface with future autonomic systems by communicating goals, priorities and tasks which the systems will solve.

The ultimate performance of a designed or operational traffic control system (e.g. urban signal control or ramp metering or variable speed limits) depends on two main factors: (a) the exogenous influences, e.g. demand, weather conditions and incidents, and (b) the values of some design parameters included in the control strategy. Every time a new control algorithm is implemented in the real world, 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 procedure is conducted manually, via trial-and-error, relying on expertise and human judgement, without the use of a systematic approach. Currently, a considerable amount of human effort and time is spent by experienced engineers, practitioners and traffic operators on tuning operational systems. In many cases, the result of this manual procedure does not lead to a desirable outcome in terms of a measurable performance metric.

Some isolated examples of autonomic properties such as self-adaptation have found their way into Intelligent Transportation Systems (ITS) and have already proved beneficial. A recently developed methodology that combines the principles of traffic engineering, automatic control, optimization and machine learning and enables online self-tuning autonomicity for operational traffic systems is presented in this chapter. This problem has been discussed in depth in [1] where the algorithm AFT (adaptive fine-tuning), which was originally introduced in [2], is analysed and tested in different simulation environments. This autonomic online procedure is aiming at replacing the conventional manual optimization practice by embedding self-tuning capabilities in control strategies. AFT can self-adjust the tunable parameters of control systems, so that they reach the maximum (measurable)

performance that is achievable with the utilized control strategy. The method can also be used for automatic readjustment of "aged" systems.

Given the positive feedback from the simulation investigations for different control problems, the algorithm is currently implemented in the field for energy-efficient building control, as well as the urban traffic control of Chania, Greece. Some preliminary results from the latter field implementation are also presented here. The results demonstrate the applicability of the algorithm and its efficiency in solving the tuning problem of real-life operational control systems.

2 Background

2.1 Problem Formulation

Consider a general discrete-time control system which is dictated by different feedback-type regulators, and its underlying dynamics are described by the following nonlinear first-order difference equation:

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

where $z(t), u_i(t), d(t)$ are the vectors of system states, control inputs and exogenous (possibly measurable) signals, respectively, $t = 0, 1, 2, \ldots$ 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 control system even if the function F is unknown.

Consider also that one or more control laws are applied to the system (1), which are described as follows:

$$u_i(t) = \varpi_i\left(\theta_i, z(t)\right) \tag{2}$$

where $\varpi_i(\cdot)$ are known smooth vector functions and θ_i is the vector of tunable parameters for the ith regulator. Note that there is no restriction imposed neither on the form of (2) nor on the number of regulators applied. Furthermore, 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):

$$\begin{aligned} J\left(\theta; z(0), D_T\right) &= \pi_T(z(T)) + \sum_{i=1}^{I}\sum_{t=0}^{T-1} \pi_{i,t}\left(z(t), u_i(t)\right) \\ &= \pi_T(z(T)) + \sum_{i=1}^{I}\sum_{t=0}^{T-1} \pi_{i,t}\left(z(t), \varpi\left(\theta_i, z(t)\right)\right) \end{aligned} \tag{3}$$

where $\theta = \text{vec}(\theta_1, \theta_2, \ldots, \theta_I)$, π_T and $\pi_{i,t}$ are known non-negative functions, I is the number of regulators that needs to be tuned, T is the finite time horizon over which the control laws (2) are applied and

$$D_T \triangleq [d(0), d(1), \ldots, d(T-1)] \tag{4}$$

denotes the time history of the exogenous signals over the optimization horizon T. By defining $x = \text{vec}(z(0), D_T)$, Eq. (3) may be rewritten as

$$J(\theta; z(0), D_T) = J(\theta, x). \tag{5}$$

AFT is an iterative algorithm which can be applied every T and will update the current system parameter vector θ so as to achieve better performance. Equation (5) indicates that the system performance depends on the vector of tunable parameters θ and the exogenous vector x. Assuming that the signal x is bounded (i.e. $|x(t)| \leq B,\ \forall\ t \in \mathbb{Z}$ for a finite value $B > 0$), it can be omitted from Eq. (5) as the objective is to optimize the expected value $E[J(\theta)]$ given the variations in x. In [3] it has been mathematically proven that the AFT algorithm asymptotically converges to the optimal solution of this problem.

The requirement for convergence itself is not sufficient in most practical implementations. Another crucial issue is the safe and efficient behaviour of the system. Algorithms similar to AFT, which enable systems with autonomic self-tuning properties, should also guarantee stable and sustainable system performance during the field deployment. The violation of this requirement in a practical application may cause serious problems (e.g. performance degradation, safety, etc.). For instance, in the case of operational traffic control systems, this could lead to serious problems (e.g. complaints, dangerous driving, etc.) that may force the traffic operators to cancel the self-tuning process. This requirement has been addressed successfully in [4] for the AFT algorithm.

2.2 Theoretical Foundations

The self-tuning problem discussed in the previous subsection is closely related to the problem of dynamic parameter estimation that has been studied for decades by many researchers. The problem of interest is to find the values of a vector $\theta^* \in \Theta$ that minimize the expected value of a scalar-valued performance function $E[J(\theta)]$ assuming that measurements of the function are available for different θ. The vector θ represents a collection of tunable (or adjustable) parameters that need to be tuned. The nonlinear function $J(\theta)$ is a scalar measure that summarizes the performance of the system and is assumed to be continuous in Θ. The vector θ^* represents the optimal solution, and the domain Θ reflects allowable values (constraints) on the components of θ and has to be a compact space.

Many stochastic approximation algorithms have been developed for the solution of this problem. Robbins and Monro [5] were the first to propose an adaptive technique for dynamic parameter estimation. Important extensions of this algorithm followed by Kiefer and Wolfowitz in [6], where the Finite Difference Stochastic Approximation (FDSA) algorithm was introduced. FDSA has provided the basis for many learning or parameter tuning algorithms in control engineering problems. An extension of FDSA is the Random Directions Stochastic Approximation (RDSA) algorithm, which was first introduced in [7] and makes use of many random perturbations of θ in order to come up with a good set of tunable parameters (based on the measurements of the performance criterion $J(\theta)$).

Finally, Spall in [8] introduced the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm for stochastic optimization of multivariate systems. This algorithm attempts to estimate the gradient $\partial J(\theta)/\partial\theta$ in one discrete time step by applying a random perturbation to the current vector θ, and it has been widely applied to parameter estimation problems. It is worth noting that SPSA does not guarantee the requirement of safe and efficient performance during the tuning process, mainly due to the use of random perturbations applied to the regulator parameters.

The theoretical intuition of AFT lies in the area of the algorithms mentioned above. However, its scope is to enable traffic control systems with autonomic self-tuning capabilities. In this chapter we explore the efficiency of AFT through simulation experiments. The problem of online self-tuning of the urban signal control strategy Traffic-responsive Urban Control (TUC) [9, 10] is investigated. A microsimulation environment of traffic flow is used for evaluating the performance of the algorithm.

3 The Self-tuning Algorithm

Figure 1 illustrates the working principle, while Table 1 denotes the variables used by the AFT algorithm. The basic functioning procedure of the self-tuning process may be summarized as follows:

- 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 tunable parameters.
- 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). Note that the performance index $J(\theta, x)$ is a (generally unknown) function of the tunable parameters θ that need to be adjusted and the main external (measurable) disturbances x (i.e. demand).
- Using the measured performance (the samples of which increase iteration by iteration), AFT 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.

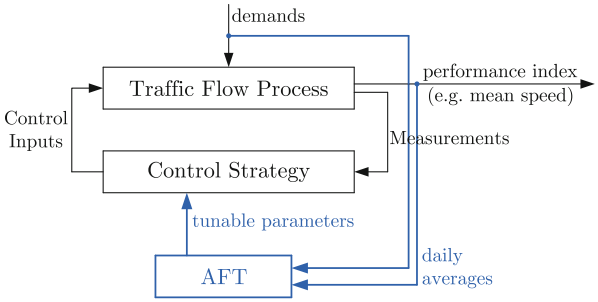

Fig. 1 Working principle of AFT for autonomic self-tuning of online traffic control systems

Table 1 Variables used within the AFT algorithm

k	Iteration index
ℓ	Past performance measurements index
J_ℓ	Performance value for the ℓth calibration experiment
$\hat{J}_\ell$	An estimate of J_ℓ obtained at the ℓth iteration
$\theta(k)$	The vector of tunable parameters at the kth calibration experiment
$\theta^*(k)$	The best set of tunable parameters until the kth experiment (according to the real measurements)
$\Delta\theta(k)^{(j)}$	Zero-mean random sequences (e.g. Gaussian), $j = 1, 2, \ldots, 2K$
$\Delta\theta(k)$	The perturbation picked by the algorithm in iteration k

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

The main components used to develop the employed algorithm are summarized as follows:

– A universal approximator $\widehat{J}(\theta, x)$ is used (e.g. a polynomial-like approximator or a neural network) in order to obtain an approximation of the nonlinear mapping $J(\theta, x)$.
– An online adaptive/learning mechanism is employed for training the above approximator. Globally convergent learning algorithms (e.g. see [11]) are required for such a purpose.
– At each algorithm iteration k, many randomly chosen candidate perturbations $\Delta\theta(k)^{(j)}$ of vector $\theta^*(k)$ are generated (where $\theta^*(k)$ is the best set of parameters so far). The effect of each candidate set $\theta^*(k) + \Delta\theta(k)^{(j)}$ to the system performance is estimated by using the approximator mentioned above. The perturbation that corresponds to the best estimate (i.e. the one that leads to the best value for $\widehat{J}(\theta^*(k) + \Delta\theta(k)^{(j)}, \hat{x}(k+1))$, where $\hat{x}$ is an estimate of the external disturbances x) is selected to determine the new values for the tunable parameters $\theta(k+1)$ to be applied at the next period (e.g. the next day).

3.1 The Universal Approximator $\hat{J}(\theta)$

The universal approximator used in the simulation experiments in order to approximate the objective function $J(\theta)$ is a linear-in-the-weights polynomial-like approximator with L_g regressor terms, which takes the form

$$\widehat{J}(\theta) = \vartheta^T \phi(\theta) \tag{6}$$

where ϑ denotes the vector of the approximator parameter estimates and

$$\phi(\theta) = \left[\phi_1(\theta), \phi_2(\theta), \ldots, \phi_{L_g}(\theta)\right]^T. \tag{7}$$

The nonlinear functions $\phi_i(\theta)$ are given by

$$\phi_i(\theta) = S^{d_1}(\theta_{m_1}) \cdot S^{d_2}(x_{m_2}),\ d_i \in \{0, 1\} \tag{8}$$

where d_i, m_i are randomly chosen at each iteration of the AFT algorithm (with $m_1, m_2 \in \{1, 2, \ldots, n_\theta\}$, where n_θ is the number of components of vector θ) and $S(\cdot)$ is a smooth monotone nonlinear function. In the neural network literature [12], this function is usually chosen to be sigmoidal. In our simulations we choose

$$S(\theta) = \tanh(\lambda_1 \theta + \lambda_2) \tag{9}$$

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

$$\min \sum_{\ell=1}^{k-1} \left(J_\ell - \vartheta^T \phi_\ell^{(k)}\right)^2. \tag{10}$$

3.2 The AFT Algorithm Description

Below, we discuss in details the application steps of the algorithm. More precisely, the steps that are executed at every iteration are as follows:

- **Step 1:** *Calculate 2K random perturbations*. In this step K random perturbations are calculated (e.g. according to Gaussian distribution). The resulting $2K$ candidate vectors $\theta^*(k) \pm \Delta\theta(k)^{(j)}$ 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 in this iteration is calculated by $L_g(k) = \min\{2(k-1), \overline{L}_g\}$ with $\overline{L}_g$ a given upper bound.

- **Step 3:** *Calculate the number of past measurements*. The algorithm keeps a window of past measurements which moves along with the iterations. T_{h} is the upper bound of the number of past measurements AFT uses and it is defined by the user. In this step the starting point of the window in the past is calculated. The end point of the window is always $k-1$.
- **Step 4:** *Produce the polynomial-like approximator*. After steps 2 and 3, the structure of the universal approximator may be formed and applied for nonlinear fitting to the data included in the window of the past measurements.
- **Step 5:** *Calculate the optimal approximator parameter estimates*. The optimal values of the approximator's parameters ϑ are calculated according to the solution of a least squares estimation method.
- **Step 6:** *Apply the* $2K$ *random perturbations* $\pm\Delta\theta(k)^{(j)}$ *to* $\widehat{J}(k)$. The $2K$ candidate vectors $\theta^*(k) \pm \Delta\theta(k)^{(j)}$ are applied to the approximator $\widehat{J}(k)$ for evaluation.
- **Step 7:** *Pick the best random perturbation (according to* $\widehat{J}(k)$*)*. The vector $\theta(k)$ with the best estimated performance is selected for application in the next simulation experiment.

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 property, which is a sufficient and necessary condition for the neural approximator $\widehat{J}(\theta)$ to be able to efficiently learn the unknown function $J(\theta)$. However, due to the use of Step 6, the proposed methodology avoids poor performance or instability problems and guarantees safe and efficient performance. As shown in [2, 3] using strict mathematical arguments, the performance of the system can be, in the worst case, similar to the system performance without the self-tuning property plus some random term. The magnitude of this term is proportional to the magnitude and variance of the exogenous inputs x.

3.3 Efficient Stochastic Stepsizes α_k

The first step of the AFT algorithm makes use of an arithmetic sequence $\alpha(k)$ which plays a critical role, often determining whether the algorithm converges or diverges. These factors are sometimes referred to as stepsizes but also gains or learning coefficients, depending on the field of application. The choice of the gain sequence $\alpha(k)$ is critical for the performance of stochastic approximation methods. In many applications, a constant stepsize is used (instead of a decaying one) as a way of avoiding gains that are too small for large k. 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 they are called stochastic stepsizes. For large-scale problems, it is possible that we have to estimate hundreds of parameters. It seems unlikely that all the parameters will

approach their optimal value at the same time (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.

Conditions guaranteeing that the stochastic approximation iterate converges to θ^* as $k \to \infty$ are presented in many places (e.g. [13]). All the existing proofs require three basic conditions about the applied stepsizes:

$$\alpha(k) \geq 0, \quad k = 0, 1, \ldots, \tag{11}$$

$$\sum_{k=0}^{\infty} \alpha(k) = \infty, \tag{12}$$

$$\sum_{k=0}^{\infty} \alpha(k)^2 < \infty. \tag{13}$$

Equation (11) requires that the stepsizes must be non-negative. The most important requirement is (12), which states that the infinite summation of stepsizes must be infinite. If this condition is violated, the algorithm might stall very early without reaching the optimal solution. Finally, condition (13) requires that the infinite sum of the squares of the stepsizes is finite. A good intuitive justification for this condition is that it guarantees that the variance of the estimate of the optimal solution goes to zero in the limit. The three conditions mentioned above provide a careful balance in having the gain $\alpha(k)$ decay neither too fast nor too slow.

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

Given the desirability for a gain sequence that balances algorithm stability in the early iterations and non-negligible gains in the later iterations, the form used in AFT is

$$\alpha_i(k) = \frac{\alpha(0)}{\alpha(0) + K_i}, \tag{14}$$

which satisfies the three conditions mentioned above.

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

$$K_i = \begin{cases} K_i & \text{if } \Delta\theta_i(k)\Delta\theta_i(k-1) > 0 \\ K_i + 1 & \text{if } \Delta\theta_i(k)\Delta\theta_i(k-1) < 0. \end{cases} \tag{15}$$

This form is inspired by the famous learning method RPROP [14]. This formula takes into account only the sign of $\Delta\theta_i$ and acts independently on each θ_i. This way, every component i of vector θ converges with a different rate according to the frequency of sign changes.

4 Simulation Experiments

In order to evaluate the efficiency of the self-tuning algorithm, extensive simulation experiments have been carried out. The microscopic simulation environment Aimsun was used in order to replicate a real-world implementation of the algorithm. In our experiments the TUC strategy [9, 10] is used to regulate the signals of the city of Chania, Greece, in real time. The system autonomously self-tunes its design parameters using the AFT algorithm. All the data utilized in Aimsun and TUC (turning rates, lost times, staging, saturation flows) are provided by the operators of the Traffic Control Centre of the city and correspond to the data of the real network. This section presents the simulation results, comparing the performance of the TUC strategy, when AFT is used for autonomic self-tuning of a predefined set of design parameters, with the base case (no AFT case). In the base case, the aforementioned parameters of the original TUC system have been manually fine-tuned to virtual perfection by the system operators [15].

4.1 Network and Simulation Setup

Chania, located at the north-western part of Crete, is the capital of the prefecture of Chania and covers 12.5 km^2. Figure 2 exhibits a satellite view of the trial urban road network (red bullets correspond to the controlled junctions), which has a total length of approximately 8 km and consists of 16 controlled junctions.

Figure 3 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 the Chania urban network links are either around the middle of the link or some 40 m upstream of the stop line. Split, cycle and offset control modules of the TUC strategy are applied to the network for all simulation investigations. For the implementation of the AFT algorithm, the following design values were used: $T_h = 90$, $\bar{L}_g = 150$, $K = 100$ and initial values to λ_i according to $\lambda_1 = 100$, $\lambda_2 = 0$. Finally, a simulation step of 0.25 s is considered for the microscopic simulation model.

Fig. 2 Satellite view of the Chania urban road network

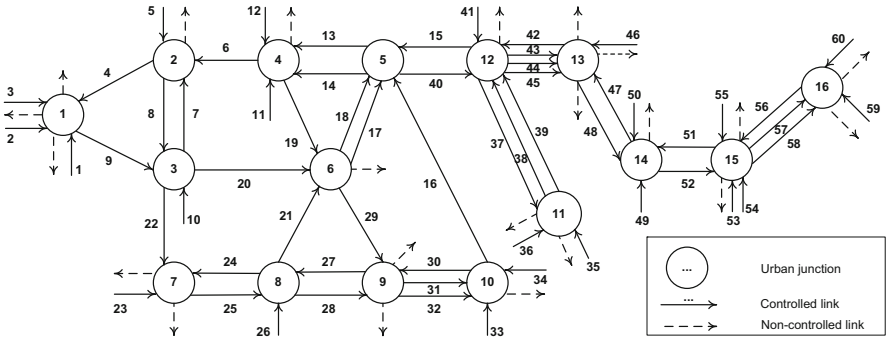

Fig. 3 Simulation model of the urban road network of Chania

4.2 Demand Scenarios and Integration with Aimsun

In order to investigate the performance of AFT 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 h. Scenario 1 comprises medium demand in all network origins, while scenario 2 comprises high demand and the network faces serious congestion for some 2 h, with some link queues spilling back into upstream links. For simplicity, we assume that a demand scenario with a time horizon of 4 h 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 the TUC strategy are updated by AFT 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 control cycle, 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 the TUC strategy to produce the traffic signal settings (splits, cycle and offsets). The signal settings are then forwarded to the microsimulator for application. The outer loop is used by AFT to update the design parameters of TUC. More specifically, at each day, Aimsun delivers the mean speed for the whole urban road network (this is the measurement of the performance index $J(\theta)$). The mean speed is used by the AFT algorithm in order to produce the new values for the design parameters of split, cycle and offset control modules of TUC (the vector θ). The new set of the design parameters is then forwarded to TUC for application and so forth.

4.3 Results from the Simulation Experiments

Table 2 presents the average results for a series of replications for the simulation of the two demand scenarios. The self-tunable system exhibits an improvement of around 17 % and 36 %, respectively. The diagrams in Fig. 4 compare the network-wide mean speed of the original TUC system (blue line) versus TUC system combined with the self-tuning algorithm (red line) for the network described above and two different demand scenarios. Scenario 1 (Fig. 4a) reflects medium demand in all network origins, while scenario 2 (Fig. 4b) reflects high demand whereby the network faces serious congestion with some link queues spilling back into upstream links. In both diagrams, it can be seen that the application of the AFT algorithm leads to better performance compared to the original TUC system. More precisely, the AFT algorithm aims to optimize the overall system performance within a few days (iteration number in x-axis), by efficiently self-tuning the design parameters for all TUC's control modules (89 parameters in total). The combined system first increases and then maintains the daily mean speed of the network at higher levels

Table 2 Comparison of the 50 days' average space-time speed (ASTS) for many simulation experiments

Demand scenario	No AFT		AFT		ASTS improvement (%)
	ASTS (km/h)	St. deviation (km/h)	ASTS (km/h)	St. deviation (km/h)	
1	16.29	0.71	19.13	0.86	17.46
2	9.67	1.63	13.19	0.82	36.33

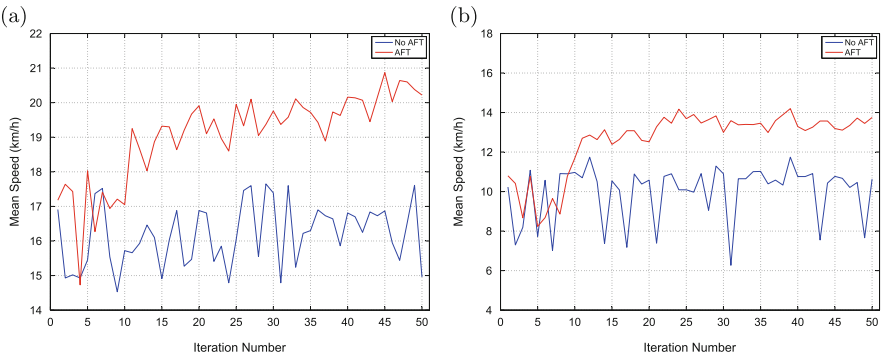

Fig. 4 Daily mean speed trajectories with and without the use of the AFT algorithm: (**a**) scenario 1, (**b**) scenario 2

than the initial day (which corresponds to the initial values of the parameters), eventually leading to a local maximum value of performance.

5 Preliminary Field Results

During the write-up period of this chapter, a field implementation and testing of the AFT algorithm took place in the city of Chania under the research project AGILE (funded by the European Commission, FP7-ICT-5-3.5). The overall system outlined in this chapter was implemented in the Traffic Control Centre of the city. More specifically, TUC was controlling the traffic signals in real time, while AFT was running in parallel, embedding self-tuning capabilities in the overall system.

Due to the fact that this is the first field experiment of the AFT algorithm to the traffic domain, some choices had to be made so as to enable a careful and gradual evaluation. The urban road network of Fig. 2 is considered for the field experiment, albeit it is separated in two regions; Region 2 consists of junctions 15 and 16 in Fig. 3, while Region 1 comprises all other junctions. Each region is controlled by its own TUC algorithm, with independent cycle times. Also, two independent versions of the AFT algorithm are running in parallel with TUC, embedding self-tuning capabilities for each region. Thus, there are two distinct experiments running at the same time. The main reason for clustering the junctions into two regions is that Region 2 is remote from the rest of the network (no offsets apply among the regions), and moreover its congestion patterns, and hence suitable cycle times, are different than for the rest of the network. It should be mentioned that traffic in Region 1, which comprises the city's CBD, is more stochastic in its behaviour, mainly due to uncertain but frequent illegal or double parking that may change the network characteristics and junction capacities in an unpredictable way.

In this first running experiment, we have chosen only four parameters for each region to be automatically tuned by the respective AFT algorithms. The four parameters determine the real-time specification of the cycle time and were found in earlier simulation-based work to influence the performance of TUC. The criterion that AFT tries to maximize is the overall mean speed. More specifically, the space-time averaged network mean speed is calculated daily based on detector data in each network link, for the period 8:00 a.m. to 2:00 p.m., which includes most of the morning and early-afternoon traffic peaks. The algorithm receives also the daily total demand (as the most important external factor for performance), which is the sum of the time-averaged (8:00 a.m. to 2:00 p.m.) flows measured by detectors at the network origins. Every day is an AFT iteration, while a total of 29 days of results (no weekends) are presented.

Figure 5 shows trajectories of parameters under AFT tuning over iterations (blue lines). The starting values of the parameters are those used in the operational system and have been manually fine-tuned in the recent past. At the beginning, some conservative bounds have been used for the parameters (red lines). The parameter of Fig. 5a is seen to hit the upper bound several times; hence we decided to change it (on the fly) as seen at iteration 15. This was also done with other parameters' upper or lower bounds.

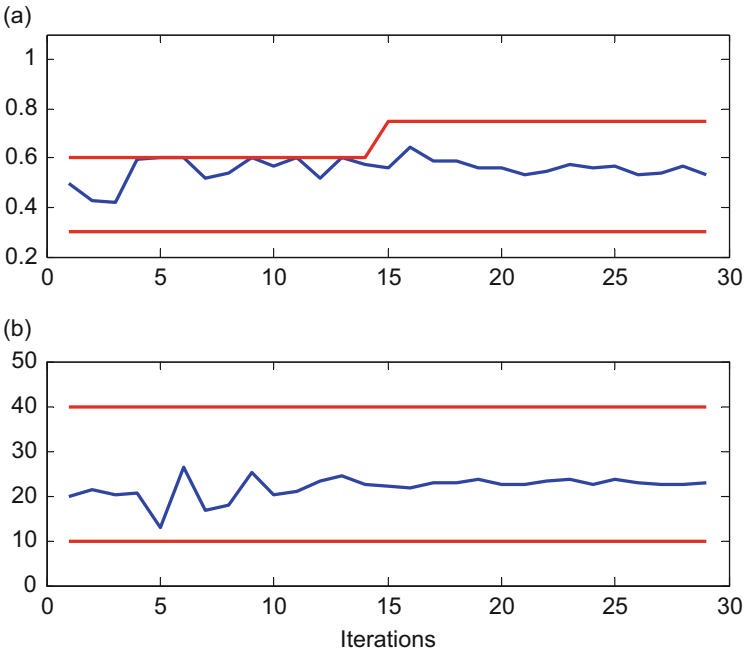

Fig. 5 Examples of trajectories of parameters (*blue*) over iterations (days) and their bounds (*red*)

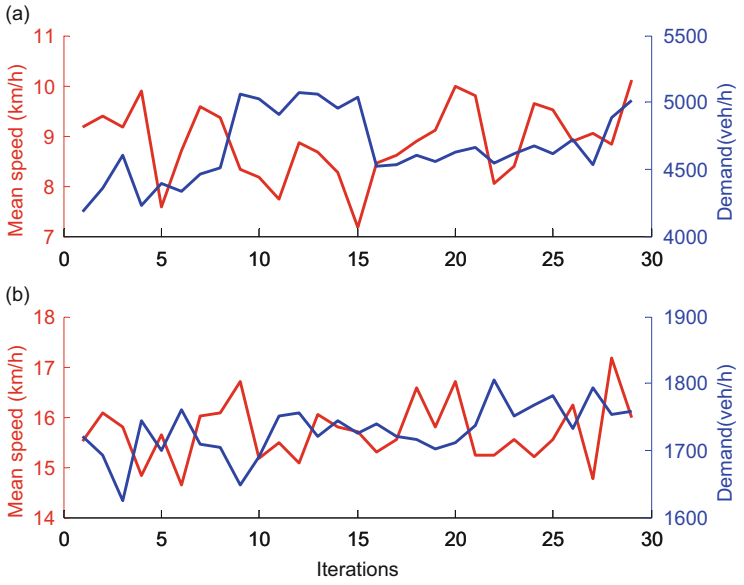

Fig. 6 Overall mean speed (*red*) and total demand (*blue*) vs. iterations: (**a**) Region 1; (**b**) Region 2

Figure 6 displays the overall mean speed and the total demand of the two networks (Region 1 and Region 2) over the available iterations. A difficulty in assessing these preliminary results is due to the fact that, in contrast to the simulation investigations, the demand is changing daily without evidence for stationary average or standard deviation. If everything else is constant, the mean speed is a decreasing function of the demand; as a result it is not possible to judge on any possible improvements by observing only the mean speed (as with the simulation results of Fig. 6). In the following, we present two ways of addressing this difficulty.

The first way is to split the available days into two groups according to their respective total demands. For Region 1, groups 1 and 2 comprise all days with demand above and below 4650 veh/h, respectively, while for Region 2, groups 1 and 2 comprise all days with demand above and below 1725 veh/h, respectively. Figure 7a, b presents the Region 1 results for the two groups, along with corresponding regression lines. It can be seen that for lower demands, there is a very slight average deterioration over iterations, while for higher demands, there is a strong improvement of about 20 % over the iterations. AFT operates for all demands and strives a total improvement. This does not exclude partial deteriorations for subdomains of demand. Figure 7c, d presents the corresponding results for Region 2, where a clear average improvement is visible for both demand groups.

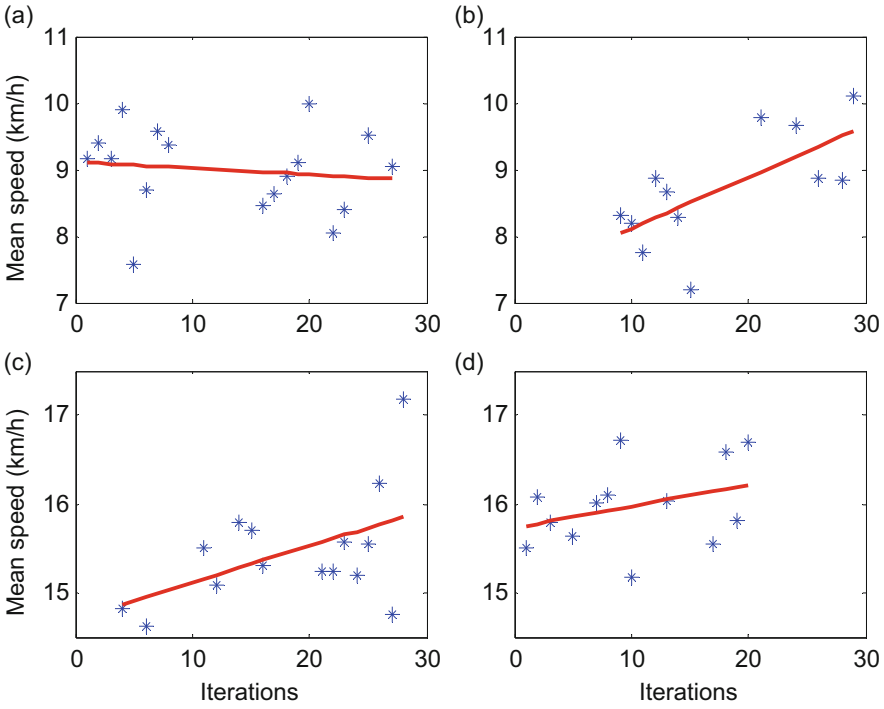

Fig. 7 (**a**) Mean speeds of Region 1 for iterations with demand lower than 4650 veh/h; (**b**) mean speeds of Region 1 for iterations with demand higher than 4650 veh/h; (**c**) mean speeds of Region 2 for iterations with demand lower than 1725 veh/h; (**d**) mean speeds of Region 2 for iterations with demand higher than 1725 veh/h

A second way to evaluate the results is by introducing a criterion which integrates the demand and mean speed. Such an evaluation criterion, which is used regularly in various studies (e.g. it is one of the three national performance criteria for Australia), is the product of the daily overall mean speed and the corresponding total demand. This is sometimes called "production" and expresses essentially the amount of veh$*$km/h^2 served by the traffic network. Figure 8 displays the evolution of this criterion over iterations for the two regions, along with the corresponding regression lines. This criterion clearly indicates that AFT achieves significant improvement to the mean speed of the network over iterations.

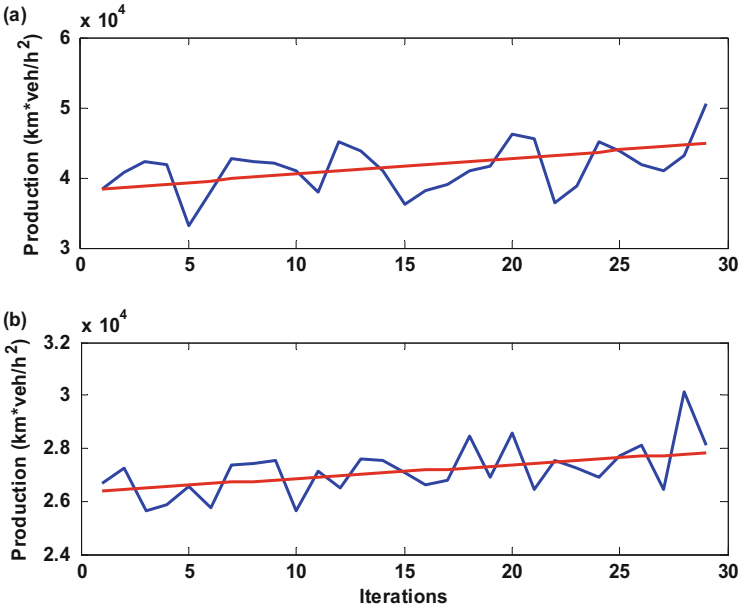

Fig. 8 Production over iterations: (**a**) Region 1 and (**b**) Region 2

6 Conclusions

This chapter investigated the efficiency of the AFT algorithm for the problem of optimizing the design parameters of traffic control systems. This adaptive optimization methodology aims at replacing the conventional manually based optimization practice with an autonomic procedure. Simulation results have been presented demonstrating that the self-tuning algorithm (AFT) leads to better network performance (in terms of daily mean speed) compared to the original TUC system. This underlines the superiority of the autonomic optimization procedure over the case where the design parameters are manually fine-tuned by field experts.

Given the conclusion of the simulation investigation, it was decided to proceed with a field implementation in the traffic network of Chania, Greece. In conclusion, the available preliminary field results of AFT are very promising and encouraging. The results confirm that despite the inhomogeneous demands, AFT evolves the control parameters so as to lead to better network average performance over time.

More field investigations have been planned in order to study in more details the impact of the strongly varying demand. One of the questions that need to be answered is the following. How many and which parameters should be selected for fine-tuning? A big set of parameters could give more degrees of freedom in the problem, but it could also lead to over-parameterization. In conclusion, the set of parameters that will be used should be the one with the highest impact

on the problem to be solved and should take into account the control strategy characteristics and of course some successful simulation results.

Acknowledgements The research leading to these results has been partially funded by the European Commission FP7-ICT-5-3.5, Engineering of Networked Monitoring and Control Systems, under the contract #257806 AGILE.

References

1. Kouvelas, A.: Adaptive fine-tuning for large-scale nonlinear traffic control systems. Ph.D. dissertation, Technical University of Crete, Chania (2011)
2. Kosmatopoulos, E.B., Papageorgiou, M., Vakouli, A., Kouvelas, A.: Adaptive fine-tuning of non-linear control systems with application to the urban traffic control strategy TUC. IEEE Trans. Control Syst. Technol. **15**(6), 991–1002 (2007)
3. Kosmatopoulos, E.B., Kouvelas, A.: Large-scale nonlinear control system fine-tuning through learning. IEEE Trans. Neural Netw. **20**(6), 1009–1023 (2009)
4. Kouvelas, A., Aboudolas, K., Kosmatopoulos, E.B., Papageorgiou, M.: Adaptive performance optimization for large-scale traffic control systems. IEEE Trans. Intell. Transp. Syst. **12**(4), 1434–1445 (2011)
5. Robbins, H., Monro, S.: A stochastic approximation method. Ann. Math. Stat. **22**, 400–407 (1951)
6. Kiefer, J., Wolfowitz, J.: Stochastic estimation of a regression function. Ann. Math. Stat. **23**, 462–466 (1952)
7. Ermoliev, Y.: On the method of generalized stochastic gradients and quasi-fejer sequences. Cybernetics **5**, 208–220 (1969)
8. Spall, J.: Multivariate stochastic approximation using a simultaneous perturbation gradient approximation. IEEE Trans. Autom. Control **37**(3), 332–341 (1992)
9. Diakaki, C., Papageorgiou, M., Aboudolas, K.: A multivariable regulator approach to traffic-responsive network-wide signal control. Control Eng. Pract. **10**, 183–195 (2002)
10. Diakaki, C., Dinopoulou, V., Aboudolas, K., Papageorgiou, M., Ben-Shabat, E., Seider, E., Leibov, A.: Extensions and new applications of the traffic-responsive urban control strategy: coordinated signal control for urban networks. Transp. Res. Rec. **1856**, 202–211 (2003)
11. Kosmatopoulos, E., Polycarpou, M., Christodoulou, M., Ioannou, P.: High-order neural network structures for identification of dynamical systems. IEEE Trans. Neural Netw. **6**(2), 422–431 (1995)
12. Huang, G.-B., Chen, L., Siew, C.-K.: Universal approximation using incremental constructive feedforward networks with random hidden nodes. IEEE Trans. Neural Netw. **17**(4), 879–892 (2006)
13. Bazaraa, M.S., Sherali, H.D., Shetty, C.M.: Nonlinear Programming: Theory and Algorithms, 2nd edn. Wiley, New York (1993)
14. Riedmiller, M., Braun, H.: RPROP – a fast adaptive learning algorithm. In: Proceedings of ISCIS VII, Universitat (1992)
15. Kosmatopoulos, E., Papageorgiou, M., Bielefeldt, C., Dinopoulou, V., Morris, R., Mueck, J., Richards, A., Weichenmeier, F.: International comparative field evaluation of a traffic-responsive signal control strategy in three cities. Transp. Res. **40A**(5), 399–413 (2006)