


A fully-autonomic methodology for embedding self-tuning competence in online traffic control systems

Conference Paper**Author(s):**

[Kouvelas, Anastasios](#) ; Kosmatopoulos, Elias B.; Papamichail, Ioannis; Papageorgiou, Markos

Publication date:

2013-10-09

Permanent link:

<https://doi.org/10.3929/ethz-b-000276243>

Rights / license:

[In Copyright - Non-Commercial Use Permitted](#)

A fully-autonomic methodology for embedding self-tuning competence in online traffic control systems

Anastasios Kouvelas¹, Elias Kosmatopoulos², Ioannis Papamichail³, Markos Papageorgiou³

Abstract—Recent advances in technology and computer science play a key role towards the design of the next generation of Intelligent Transportation Systems (ITS). The architecture of such complex systems is crucial to include supporting algorithms that can embody self managing properties within the existing ITS strategies. This paper presents a recently developed adaptive optimization algorithm that combines methodologies from the fields of traffic engineering, optimization and machine learning in order to embed self-tuning properties in traffic control systems. The derived Adaptive Fine-Tuning (AFT) algorithm comprises a fully-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 is evaluated in simulation experiments, examining the ability of self-tuning the design parameters of a traffic control strategy for urban road networks.

I. INTRODUCTION

Despite the continuous advances in the fields of control and computing, the design and deployment of efficient traffic control systems 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 for the system dynamics, as the use of more complex models is virtually unavoidable in most real life applications. As a result, although the derived regulators are theoretically optimal, they usually exhibit suboptimal performance. The ultimate performance of a designed or operational traffic control system (e.g. urban signal control or ramp metering or variable speed limit) 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 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 judgment and 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.

This paper presents a recently developed methodology that combines the principles of traffic engineering, automatic control and machine learning and enables online self-tuning autonomicity to operational traffic systems. This problem is discussed in depth in [1] where the algorithm AFT (Adaptive Fine-Tuning), which was originally introduced in [2], is analyzed and tested in different simulation environments. This fully 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.

II. BACKGROUND

A. 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

$$z(t+1) = F(z(t), u_i(t), d(t), t), \quad z(0) = z_0 \quad (1)$$

where $z(t)$, $u_i(t)$, $d(t)$ are the vectors of system states, control inputs, and exogenous (possibly measurable) signals, respectively, $t = 0, 1, 2, \dots$ 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

$$u_i(t) = \varpi_i(\theta_i, z(t)) \quad (2)$$

where $\varpi_i(\cdot)$ are known smooth vector functions and θ_i is the vector of tunable parameters for the i -th regulator. Note, that there is no restriction imposed neither on the form of (2) nor on the number of the regulators applied. Furthermore, the discrete time-index t may be different for each control law i .

¹Anastasios Kouvelas is with Partners for Advanced Transportation Technology, University of California at Berkeley, 94720, CA, USA, kouvelas@berkeley.edu

²Elias Kosmatopoulos is with the Department of Electrical and Computer Engineering, Democritus University of Thrace, 67100, Xanthi, Greece, kosmatop@ee.duth.gr

³Ioannis Papamichail and Markos Papageorgiou are with the Department of Production and Management Engineering, Technical University of Crete, 73100, Chania, Greece, {ipapa, markos}@dssl.tuc

The overall system performance is evaluated through the following objective function (performance index)

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

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

$$D_T \triangleq [d(0), d(1), \dots, d(T-1)] \quad (4)$$

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

$$J(\theta; z(0), D_T) = J(\theta, x). \quad (5)$$

AFT is an iterative algorithm which can be applied every T and will update the current system parameters 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 equation (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 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 behavior of the system. Algorithms similar to AFT, that 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 AFT algorithm.

B. 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 FDSA (Finite Difference Stochastic Approximation) 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 RDSA (Random Directions Stochastic Approximation) algorithm, which was firstly 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 SPSA (Simultaneous Perturbation Stochastic Approximation) 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 algorithm 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 paper we explore the efficiency of AFT through simulation experiments. The problem of online self-tuning of the urban signal control strategy TUC ([9], [10]) is investigated. A micro-simulation environment of traffic flow is used for evaluating the performance of the algorithm.

III. THE SELF-TUNING ALGORITHM

A. Introduction

Fig. 1 illustrates the working principle of 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), 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 $J(\theta)$ is a (generally unknown) function of the tunable parameters θ that need to be adjusted.
- Using the measured performance (the samples of which increase iteration by iteration), 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.

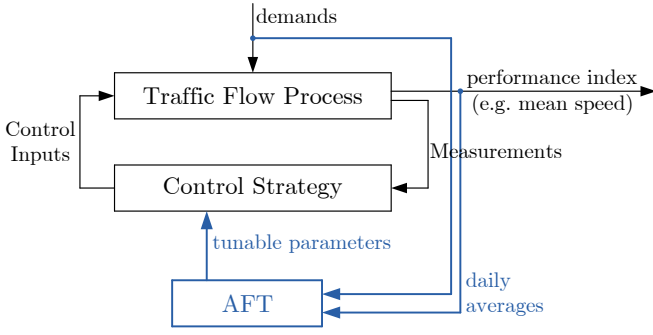


Fig. 1. Working principle of AFT for autonomous self-tuning of online traffic control systems.

- This (iterative) procedure is continued over many periods (e.g. days) until a maximum in performance is reached; then, 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).

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

- A universal approximator $\hat{J}(\theta)$ is used (e.g., a neural network or a polynomial-like approximator) in order to obtain an approximation of the nonlinear mapping $\hat{J}(\theta) \equiv J(\theta)$.
- An on-line adaptive/learning mechanism is employed for training the above approximator. Globally convergent learning algorithms (see e.g., [11], [12]) 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 of the candidate sets $\theta^*(k) + \Delta\theta(k)^{(j)}$ 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 $\hat{J}(\theta^*(k) + \Delta\theta(k)^{(j)})$) is picked to depict the new values for the tunable parameters $\theta(k+1)$. This vector is selected to be applied at the next period (e.g. the next day).

B. 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

$$\hat{J}(\theta) = \vartheta^T \phi(\theta) \quad (6)$$

where ϑ denotes the vector of the approximator parameter estimates and

$$\phi(\theta) = [\phi_1(\theta), \phi_2(\theta), \dots, \phi_{L_g}(\theta)]^T. \quad (7)$$

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

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

where d_i, m_i are randomly chosen at each iteration of AFT algorithm (with $m_1, m_2 \in \{1, 2, \dots, n_\theta\}$, where n_θ is the number of components of vector θ) and $S(\cdot)$ is a smooth monotone nonlinear function. In the neural networks literature [13], [14] this function is usually chosen to be sigmoidal. In our simulations we choose

$$S(\theta) = \tanh(\lambda_1 \theta + \lambda_2) \quad (9)$$

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

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

C. AFT algorithm description

Below, we discuss in details the application steps of the algorithm. Table I denotes the variables used while Table II displays all the steps that are performed at every iteration k 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 (according to e.g. 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.
- **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 $k-1$.
- **Step 4: Produce the polynomial-like approximator.** After steps 2, 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 $\hat{J}(k)$.** The $2K$ candidate vectors $\theta^*(k) \pm \Delta\theta(k)^{(j)}$ are applied to the approximator $\hat{J}(k)$ for evaluation.
- **Step 7: Pick the best random perturbation (according to $\hat{J}(k)$).** 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 $\hat{J}(\theta)$ to be able to efficiently learn the unknown

TABLE I
VARIABLES USED WITHIN AFT ALGORITHM

k	iteration index
ℓ	past performance measurements index
J_ℓ	performance value for the ℓ -th calibration experiment
\hat{J}_ℓ	an estimate of J_ℓ 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 (according to the real measurements)
$\Delta(k)^{(j)}$	zero-mean random sequences (e.g. Gaussian), $j = 1, 2, \dots, 2K$
$\Delta\theta(k)$	the perturbation picked by the algorithm in iteration k

TABLE II
AFT ALGORITHM DESCRIPTION

1) calculate $2K$ random perturbations $\Delta\theta(k)^{(j)} = \alpha(k)\Delta(k)^{(j)}, j \in \{1, 2, \dots, 2K\}$
2) calculate the number of approximator regressor terms $L_g(k) = \min\{2(k-1), L_g\}$
3) calculate the number of past measurements $\ell(k) = \max\{k - T_h, 1\}$
4) produce the polynomial-like approximator $\phi_\ell(k) = \phi(\theta_\ell(k))$
5) calculate the optimal approximator parameter estimates $\vartheta(k) \mapsto \arg \min_{\vartheta(k)} \frac{1}{2} \sum_{\ell=\ell(k)}^{k-1} (J_\ell - \vartheta^\tau \phi_\ell(k))^2$
6) apply the $2K$ random perturbations $\pm\Delta\theta(k)^{(j)}$ to $\hat{J}(k)$ $\hat{J}(\pm\Delta\theta(k)^{(j)} + \theta^*(k)) = \vartheta(k)^\tau \phi(\pm\Delta\theta(k)^{(j)} + \theta^*(k))$
7) pick the best random perturbation (according to $\hat{J}(k)$) $\Delta\theta(k) = \arg \max_{\Delta\theta(k)^{(\pm j)}} \hat{J}(\pm\Delta\theta(k)^{(j)} + \theta^*(k))$
$\alpha(k)$ is a user-defined positive sequence (e.g. constant stepsize $\alpha(k) \equiv \alpha \in (0, 1)$)
T_h, L_g, K are user-defined positive integers
$\theta^*(k) + \Delta\theta(k)$ denotes the vector of tunable parameters picked to be applied at the next experiment $k+1$

function $J(\theta)$. However, due to the use of Step 6 (see Table II) 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 .

D. Efficient stochastic stepsizes α_k

The first step of AFT algorithm (Table II) 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 $k \rightarrow \infty$ are presented in many places (e.g., [15], [16]). All the existing proofs require three basic conditions about the applied stepsizes

$$\alpha(k) \geq 0, \quad k = 0, 1, \dots, \quad (11)$$

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

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

Equation (11) requires that the stepsizes must be nonnegative. 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. Figure 2 presents the decaying sequence

$$\alpha(k) = \frac{\alpha(0)}{\alpha(0) + k} \quad (14)$$

which satisfies the three conditions mentioned above, for $k = 1, 2, \dots, 50$ and different values of parameter $\alpha(0)$.

For our experiments, we introduce an adaptive technique for the calculation of stepsize $\alpha_i(k)$ (where i refers to the i -th component of vector θ), at each iteration of 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

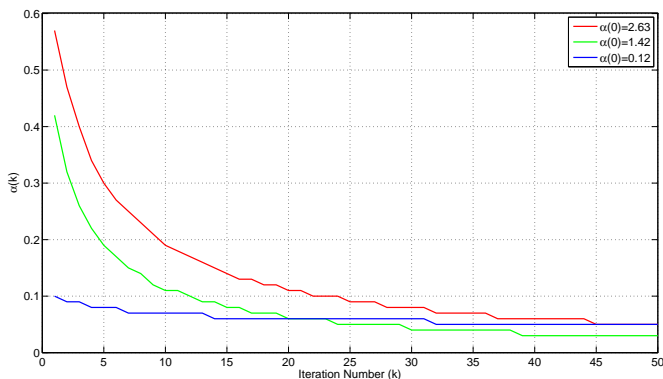


Fig. 2. Evolution of stepsizes for 50 iterations when using decaying sequence $\alpha(k) = \frac{\alpha(0)}{\alpha(0)+k}$ for three different values of $\alpha(0)$.

gains in the later iterations, the form used in AFT is

$$\alpha_i(k) = \frac{\alpha(0)}{\alpha(0) + K_i}. \quad (15)$$

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

$$K_i = \begin{cases} K_i & \text{if } \Delta\theta_i(k)\Delta\theta_i(k-1) > 0 \\ K_i + 1 & \text{if } \Delta\theta_i(k)\Delta\theta_i(k-1) < 0. \end{cases} \quad (16)$$

This form is inspired by the famous learning method RPROP [17] and the arithmetic sequence (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. Another common technique is to periodically restart the arithmetic sequence. That is, after starting the search with an initial condition $\theta(0)$, $\alpha(0)$, periodically restart the gain sequence at $\alpha(0)$.

IV. 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 Traffic-responsive Urban Control (TUC) strategy is used to regulate the signals of the city of Chania, Greece, in real-time. The system autonomously self-tunes its design parameters by the use of AFT algorithm. All the data used in Aimsun and TUC (turning rates, lost times, staging, saturation flows) are provided by the operators of the Traffic Control Center of the city and correspond to the data of the real network. This paper presents the simulation results, comparing the performance of TUC strategy when using AFT 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 [18].

A. Brief Introduction to TUC strategy

TUC ([9], [10]) is a recently developed, efficient real-time urban traffic control strategy. Its design principles are based on feedback control theory as opposed to most of the existing strategies employing model-based optimization techniques. TUC consists of three distinct interconnected control modules that allow for real-time control of the following:

- **Green times (splits):** This module splits the total effective green time duration to the different stages of an intersection. The control objective is to minimize the risk of oversaturation and the spillback of link queues. For accomplishing this, the strategy is suitably varying, in a coordinated manner, the green-phase duration of all stages at all network junctions around some nominal values. The changes of the splits are done in real-time without affecting the offsets or cycle times. A multivariable regulator is used in this module which is derived using the Linear-Quadratic control theory (see [9], [10] for details).
- **Cycle time:** One way to influence traffic conditions via traffic lights is by modifying cycle time. Fundamentally, a longer cycle time typically increases the junction capacity, because the proportion of the constant lost time (intergreen) 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 these remarks, 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. 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 critical links ([10]).
- **Offsets:** The objective of this module is to coordinate the traffic lights of successive intersections and achieve a green wave along an arterial. TUC performs offset control in a decentralized way, that is, for successive couples of junctions along the predefined 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 possible existence of vehicle queues while specifying the offset between two successive junctions, through the application of a simple decentralized feedback control law.

These three control modules are complemented by a fourth module for provision of public transport priority. All control modules are based on feedback concepts of various types, which leads to TUC's computational simplicity as compared to model-based optimization approaches, without actually sacrificing efficiency. In this paper, we will concentrate on the autonomic tuning of TUC's split, cycle and offset control



Fig. 3. Satellite view of the Chania urban road network.

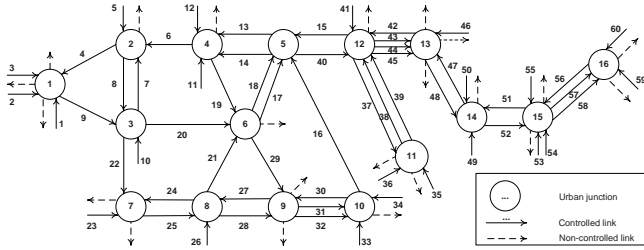


Fig. 4. Simulation model of the Chania urban road network.

module parameters.

B. Network and simulation setup

Chania, located at the north-western part of Crete, is the capital of the prefecture of Chania and covers 12.5km². Chania is the second biggest prefecture of Crete in size, population and development. Figure 3 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.

Figure 4 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. Split, cycle and offset control modules of TUC strategy are applied to the network for all simulation investigations. For the implementation of 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 0.25s is considered for the microscopic simulation model.

C. 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 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 TUC strategy to produce the traffic signal settings (splits, cycle, and offsets). The 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 (this is the measurement of the performance index $J(\theta)$). The mean speed is used by AFT algorithm in order to produce the new values for the design parameters of split, cycle and offset control modules of TUC strategy (the vector θ). The new set of the design parameters is then forwarded to TUC strategy for application, and so forth.

V. EVALUATION RESULTS

This section presents the quantitative results of the simulation experiments. It should be noted that Aimsun is a stochastic model, which implies that it 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 in order to account for the statistical variations of the results.

A. Results for demand scenario 1

The simulation results obtained for demand scenario 1 described above (medium demand) are presented here. Table III displays the mean speed for the original TUC system and the mean speed when using AFT for the system self-tuning, for each replication. Also, the standard deviation of the mean speed due to the $\pm 5\%$ 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. Table III also displays the number of days needed for AFT to converge (column 4). The evaluation results of AFT (i.e. the mean speed for the whole network, column 5 in Table III) are computed for the simulation horizon after the convergence of the algorithm (i.e. after the convergence day). Thus, the first iterations of the learning procedure

TABLE III

COMPARISON OF THE AVERAGE MEAN SPEED (MS) OF TUC STRATEGY WITH AND WITHOUT THE SELF-TUNING ALGORITHM (AFT) FOR DEMAND SCENARIO 1.

ID	No AFT		Conv. Day	AFT (after convergence day)		
	MS	St.Dev.		MS	St.Dev.	Improvement
1	16.41	0.68	10	20.04	0.83	22.10%
2	16.53	0.80	9	19.51	1.02	18.02%
3	16.09	0.51	4	19.85	0.71	23.43%
4	16.30	0.56	8	19.52	0.90	19.72%
5	16.40	0.66	5	18.95	1.00	15.55%
6	16.29	0.73	9	19.25	0.97	18.18%
7	16.38	0.73	6	19.26	0.91	17.59%
8	16.23	0.86	14	19.32	0.65	19.02%
9	16.07	0.68	8	19.79	0.88	23.15%
10	16.18	0.90	11	19.53	0.73	20.69%
Average	16.29	0.71	8.4	19.50	0.86	19.73%

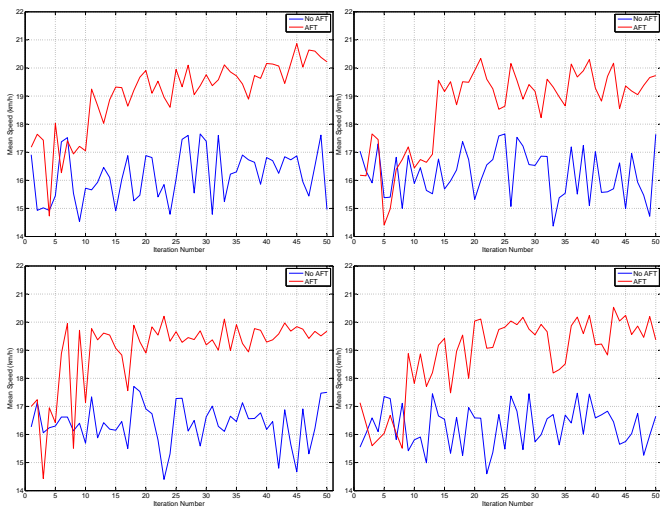


Fig. 5. Mean speed trajectories with and without the use of AFT algorithm for scenario 1 (4 different replications).

are excluded from the mean speed comparison. Finally, the percentage of improvement over the no AFT case is shown in the last column of the table.

It can be seen that the use of the self-tuning algorithm leads to an average improvement of the system performance of some 20% for this demand scenario. 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 the self-tuning process this speed is increased to 19.50 km/h.

Diagrams in Fig. 5 compare the network-wide mean speed of the original TUC system (blue line) versus TUC system combined with the self-tuning algorithm (red line) delivered for scenario 1. In all diagrams, 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 x-axis), by efficiently self-tuning the design parameters for all TUC's control modules.

At the same time, the overall system maintains the daily mean speed of the network always in higher levels than the initial day (which corresponds to the initial values of the parameters).

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 graphs are due to the $\pm 5\%$ daily random perturbations applied to the demand scenario. In all simulation runs, equations (15)–(16) were applied for the stepsize of AFT algorithm, with $\alpha(0) = 1.42$. The decaying sequence of $\alpha(k)$ provided by this formula has the advantage that after some iterations converges to small values for the stepsize $\alpha(k)$, providing the algorithm with candidate sets of parameters close to the optimal solution found so far.

B. Results for demand scenario 2

This section presents the simulation results obtained for the demand scenario 2 (high demand). Table IV displays the mean speed for the original TUC system and the mean speed when using AFT for the system self-tuning, for each replication. Also, the standard deviation of the mean speed due to the $\pm 5\%$ 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 (column 4 in Table IV) are computed after the convergence of the algorithm (column 3). Finally, the last column of Table IV displays the percentage of 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 41% for this demand scenario, while the time needed for convergence is 7.9 days on average. 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.

Diagrams in Fig. 6 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 diagrams, 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 x-axis), 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 graphs, 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 diagrams (no AFT case), are due to the $\pm 5\%$ daily random perturbations applied to the demand scenario. The same perturbations are also applied

TABLE IV

COMPARISON OF THE AVERAGE MEAN SPEED (MS) OF TUC STRATEGY WITH AND WITHOUT THE SELF-TUNING ALGORITHM (AFT) FOR DEMAND SCENARIO 2.

ID	No AFT		Conv. Day	AFT (after convergence day)		
	MS	St.Dev.		MS	St.Dev.	Improvement
1	9.99	1.41	12	13.33	0.44	33.35%
2	10.39	1.36	5	13.41	1.25	29.03%
3	9.87	1.60	8	13.49	0.91	36.67%
4	9.90	1.56	6	13.55	1.18	36.91%
5	9.97	1.79	15	14.37	0.55	44.14%
6	9.19	1.88	6	13.85	0.42	50.61%
7	8.95	1.92	4	13.22	0.77	47.74%
8	9.06	1.56	4	13.14	0.76	44.93%
9	10.04	1.54	13	13.86	0.90	38.09%
10	9.39	1.67	6	13.93	1.06	48.31%
Average	9.67	1.63	7.9	13.61	0.82	40.70%

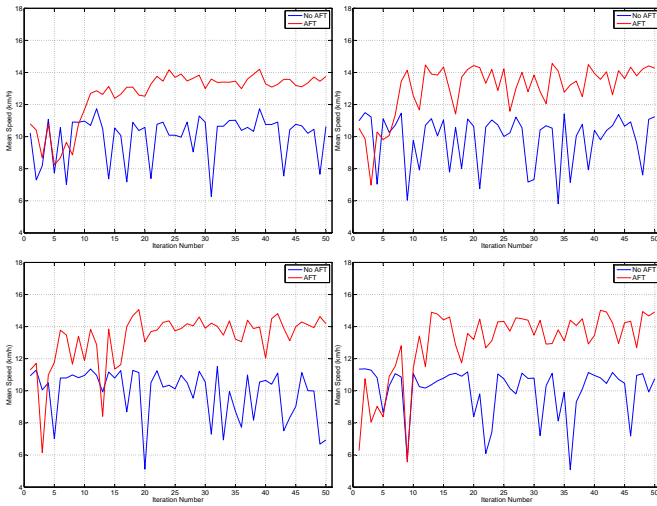


Fig. 6. Mean speed trajectories with and without the use of AFT algorithm for scenario 2 (4 different replications).

to the AFT case. However, the oscillations of the red lines (after the convergence of the algorithm) are clearly lower. Finally, the diagrams 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.

VI. CONCLUSIONS

The paper investigated the efficiency of 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 a fully-autonomic procedure. Extensive simulation experiments have been conducted and it has been demonstrated 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 fully-autonomic optimization procedure over the case where the design parameters are manually fine-tuned by field experts.

Currently, there is an ongoing field implementation taking place in the city of Chania under the research project AGILE (funded by European Commission FP7-ICT-5-3.5, Engineering of Networked Monitoring and Control Systems, contract #257806). The overall system described in this paper is implemented in the Traffic Control Center of the city. TUC strategy is controlling the traffic lights in real-time and AFT algorithm is running in parallel embedding self-tuning capabilities in the control strategy. The results of this study are going to demonstrate if the algorithm is feasible and efficient in real life conditions.

REFERENCES

- [1] A. Kouvelas, "Adaptive fine-tuning for large-scale nonlinear traffic control systems," Ph.D. Dissertation, Technical University of Crete, Chania, Greece, 2011.
- [2] 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*, vol. 15, no. 6, pp. 991–1002, 2007.
- [3] E. B. 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.
- [4] A. Kouvelas, K. Aboudolas, E. B. Kosmatopoulos, and M. Papageorgiou, "Adaptive performance optimization for large-scale traffic control systems," *IEEE Transactions on Intelligent Transportation Systems*, 2011.
- [5] H. Robbins and S. Monro, "A stochastic approximation method," *Annals of Mathematical Statistics*, vol. 22, pp. 400–407, 1951.
- [6] J. Kiefer and J. Wolfowitz, "Stochastic estimation of a regression function," *Annals of Mathematical Statistics*, vol. 23, pp. 462–466, 1952.
- [7] Y. Ermoliev, "On the method of generalized stochastic gradients and quasi-fejer sequences," *Cybernetics*, vol. 5, pp. 208–220, 1969.
- [8] J. Spall, "Multivariate stochastic approximation using a simultaneous perturbation gradient approximation," *IEEE Transactions on Automatic Control*, vol. 37, no. 3, pp. 332–341, 1992.
- [9] C. Diakaki, M. Papageorgiou, and K. Aboudolas, "A multivariable regulator approach to traffic-responsive network-wide signal control," *Control Engineering Practice*, vol. 10, pp. 183–195, 2002.
- [10] 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*, no. 1856, pp. 202–211, 2003.
- [11] E. Kosmatopoulos, M. Polycarpou, M. Christodoulou, and P. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 422–431, 1995.
- [12] E. Kosmatopoulos, M. Christodoulou, and P. Ioannou, "Dynamical neural networks that ensure exponential identification error convergence," *Neural Networks*, vol. 10, no. 2, pp. 299–314, 1997.
- [13] V. Maiorov and R. Meir, "Approximation bounds for smooth functions in $C(\mathbb{R}^d)$ by neural and mixture networks," *IEEE Transactions on Neural Networks*, vol. 9, no. 5, pp. 969–978, 1998.
- [14] 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*, vol. 17, no. 4, pp. 879–892, 2006.
- [15] M. S. Bazaraa, H. D. Sherali, and C. M. Shetty, *Nonlinear Programming: Theory and Algorithms*, 2nd ed. New York: John Wiley & Sons, Inc., 1993.
- [16] W. B. Powell, *Approximate Dynamic Programming: Solving the Curses of Dimensionality*. Princeton, N.J.: John Wiley, 2007.
- [17] M. Riedmiller and H. Braun, "RPROP – A fast adaptive learning algorithm, Proceedings of ISICIS VII, Universitat," 1992.
- [18] 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*, vol. 40A, no. 5, pp. 399–413, 2006.