Adaptive fine-tuning of conlinear control systems with application to the urban traffic control strategy TUC

Author(s):
Kosmatopoulos, Elias B.; Papageorgiou, Markos; Vakouli, Antigoni; Kouvelas, Anastasios

Publication Date:
2007-11

Permanent Link:

https://doi.org/10.3929/ethz-b-000275939 →

Originally published in:

IEEE Transactions on Control Systems Technology 15(6), http://doi.org/10.1109/tcst.2007.899645 →

Adaptive Fine-Tuning of Non-linear Control Systems with Application to the Urban Traffic Control Strategy TUC

E. B. Kosmatopoulos, M. Papageorgiou *Fellow, IEEE*, A. Vakouli, A. Kouvelas

***Abstract*— Practical large-scale nonlinear control systems require an intensive and time-consuming effort for the fine-tuning of their control parameters in order to achieve a satisfactory performance. In most cases, the fine-tuning process may take years and is performed by experienced personnel. The purpose of this paper is to introduce and analyze a systematic approach for the automatic fine-tuning of the control parameters of practical large-scale nonlinear control systems and investigate its efficiency when applied to the recently developed urban traffic control strategy TUC (Traffic-responsive Urban Control). The proposed approach is based on a concept similar to the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm. The difference between the SPSA algorithm and the proposed approach is that, SPSA employs an approximation of the gradient of an appropriate objective function using only the most recent fine-tuning experiments, while in the proposed approach the approximation of the gradient is performed by using a linear-in-the-parameters approximator that incorporates information of a user-specified time-window of the past experiments. Mathematical analysis of the proposed approach establishes its convergence properties and that SPSA can be regarded as a special case of the proposed approach. Simulation results using the traffic network of the city of Chania, Greece - a typical urban traffic network containing all possible varieties of complex junction staging - demonstrate the efficiency of the proposed approach.**

***Index Terms*— Adaptive Optimization, Stochastic Approximation, Simultaneous Perturbation Stochastic Approximation (SPSA), Nonlinear Control, Adaptive Fine-Tuning**

I. Introduction

A. Motivation

The motivation for the adaptive approach presented in this paper was the need for the development of an automatic fine-tuning procedure for the parameters of the recently developed urban traffic signal control strategy TUC (Traffic-responsive Urban Control) [5], [13], [6]. TUC is an efficient real-time traffic control strategy whose design principles are based on feedback control theory as opposed to most of the existing strategies employing model-based optimization techniques. Under the European research project SMARTNETS, TUC has been implemented in three traffic networks with quite different traffic and control infrastructure characteristics: Chania, Greece (23 junctions); Southampton, U.K. (53 junctions); and Munich, Germany (25 junctions), where it has been compared to the respective resident real-time traffic control strategies TASS, SCOOT and BALANCE [13], [2]. The main conclusions drawn from this comparative evaluation are that TUC is an easy-to-implement, inter-operable, low-cost real-time traffic control strategy whose performance, after limited fine-tuning, proved to be better or, at least, similar to the ones achieved by the resident long-standing strategies that were in most cases very well fine-tuned over the years in the specific networks. More specifically, while the TUC parameter fine-tuning took place in a period less than 1 month, the residence strategies' good performance has been achieved after extensive fine-tuning over many years. In all three networks, the fine-tuning of TUC parameters was performed by experienced personnel based on field observations and by experimenting with different combinations of parameter values, without the use of a systematic approach.

The lack of a systematic approach in the fine-tuning of TUC parameters motivated us to investigate the possibility of using adaptive optimization techniques; a very promising and quite popular technique is the Simultaneous Perturbation Stochastic Approximation (SPSA) technique [16], [18]. SPSA is a computationally efficient stochastic approximation technique that overcomes the problem emerging in conventional stochastic approximation algorithms where the number of function evaluations needed for each algorithm iteration is proportional to the dimension of the parameter vector, thus, rendering their application to high-dimensional problems impossible. SPSA has been efficiently implemented in the fine-tuning of controller parameters in many control applications (see e.g., [19]); it is worth noticing that the SPSA technique has been also applied to the fine-tuning of the parameters of a neural network-type urban traffic control system [17]. The efficiency of the technique has been demonstrated in [17] through simulations of part of the traffic network of the central business district of Manhattan, New York.

Unfortunately, the application of SPSA to the fine-tuning of TUC parameters failed to produce satisfactory results for the case of TUC's application to the traffic network of the city of Chania, Greece. It is worth noticing that, while the junction staging structure of the Manhattan network simulated in [17] is a very simple one (two stages per junction), the junction staging structure in Chania (and most urban networks in Europe) is more complex, containing all possible varieties of junction staging (e.g., double junctions with common staging, junctions with 3 and more stages, junctions where phases receive green in more than one stages, etc).

Manuscript received May 15, 2006; revised October 6, 2006.

The authors are with Dynamic Systems and Simulation Laboratory, Department of Production & Management Engineering, Technical University of Crete, Chania 73100, Greece, tel: +30-28210-37306, email: kosmatop@dssl.tuc.gr

The failure of SPSA to produce satisfactory results motivated us to develop a new algorithm, the Adaptive Fine-Tuning (AFT). As in the case of SPSA, the AFT algorithm uses a small number of objective function evaluations for each algorithm iteration. Moreover, as in the SPSA case, the objective function evaluations at each iteration of the AFT algorithm, are performed on random perturbations around the current control parameter vector. The main difference between both algorithms is that SPSA uses the finite differences of these function evaluations to estimate the gradient of the objective function, while in the AFT case a Linear-in-the-weights Universal Approximator (LUA) is used to locally approximate the objective function as a function of the control parameters; the gradient of the objective function is then approximated by the gradient of the LUA. Using mathematical analysis it is shown that AFT preserves similar convergence properties as SPSA; the convergence proof of the AFT algorithm also reveals its superiority over SPSA. More specifically, it is shown that the SPSA algorithm can be regarded as a special case of the AFT algorithm if a specific simple LUA is used.

Simulation results using a macroscopic simulation model of the complex traffic network of the city of Chania, demonstrate the efficient performance of the proposed approach. Contrary to SPSA, the proposed approach manages to improve significantly TUC's performance even in the case of high daily traffic demand variation and in the case of little or zero knowledge about the traffic network characteristics. The improvement of TUC's efficiency is significant even in the case where TUC is designed based on the assumption of perfect knowledge of traffic network characteristics.

The simulation results concentrate only on the fine-tuning of the split control module of TUC, which is indeed the most difficult to fine-tune. We note that very useful conclusions are drawn from the simulations regarding the significance of particular control parameters to the efficiency of TUC and the effect of fine-tuning to these parameters.

B. Preliminaries

A function f is said to be C^m, where m is a positive integer, if it is continuous and its first m derivatives are continuous. If $f_\alpha(\cdot)$ is a function parametrized by the nonnegative vector α, we say that $f_\alpha(\cdot)$ is $\mathcal{O}(\alpha)$, if there exists a scalar function $g(\cdot)$ satisfying $g(0)=0, g(\alpha)>0,\ \forall \alpha \neq 0$, and

$$g\left([\alpha_1,\alpha_2,\ldots,\alpha_i,\ldots,\alpha_n]^\tau\right) < g\left([\alpha_1,\alpha_2,\ldots,\alpha_i+\Delta\alpha,\ldots,\alpha_n]^\tau\right),\ \forall i,\ \forall \Delta\alpha>0$$

such that for every x and every $\alpha \geq 0$ there exists a nonnegative real c_1 for which $|f_\alpha(x)| \leq c_1 g(\alpha)$.

The notation $\mathrm{vec}\,(A,B)$, where A, B are vectors or matrices, is used to denote a vector whose elements are the entries of A and B (taken columnwise). If x is a vector then $\mathrm{diag}(x_i)$ denotes the diagonal matrix whose diagonal entries are the entries of x.

C. Linear-in-the-Weights Universal Approximators

The form of a Linear-in-the-Weights Universal Approximator (LUA) is as follows

$$y = \vartheta^\tau \phi(x) \tag{1.1}$$

where $x \in \Re^{n_1}$ denotes the input vector to the LUA, $y \in \Re^{n_2}$ denotes the output vector of the LUA, $\vartheta \in \Re^{n_2 \times L}$ denotes a matrix of constant parameters , $\phi : \Re^{n_1} \mapsto \Re^L$ is a nonlinear vector function of *regressor terms* and the integer L denotes the number of regressor terms. It can be shown that various neural network models, fuzzy systems, polynomial approximators belong to the class (1.1); for instance high order neural networks, radial basis function networks, neural network with shifted sigmoidals, adaptive fuzzy systems, etc, (see e.g. [3], [4], [15], [14], [1], [10], [11], [12] and the references therein) belong to the class of LUA (1.1).

A very important property that many LUA of the form (1.1) satisfy is the following:

(P1) We say that a family of LUA of the form (1.1) is a family of universal approximators, if for every continuous function $F : \Re^{n_1} \mapsto \Re^{n_1}$, for every $\varepsilon > 0$ and every compact subset $\mathcal{X} \subset \Re^{n_1}$ there is an integer L and a matrix ϑ^* such that the LUA with L regressor terms satisfies

$$\sup_{x\in\mathcal{X}} |F(x) - \vartheta^{*\tau}\phi(x)| \leq \varepsilon$$

Many families of LUA of the form (1.1) such as high order neural networks, radial basis function networks, neural network with shifted sigmoidals satisfy property (P1).

Let us now fix L, the function $F(\cdot)$ and the compact set $\mathcal{X}$. Then, the *optimal parameter matrix* ϑ^* and the *optimal modeling error* $\nu(\cdot)$ w.r.t. $L, \phi, F(\cdot)$ and $\mathcal{X}$ are defined as follows: $\vartheta^* \triangleq \mathcal{W}(L,\phi,F,\mathcal{X}) \triangleq \arg\min_\vartheta \sup_{x\in\mathcal{X}} |F(x) - \vartheta^{*\tau}\phi(x)|$ and $\nu(x) \triangleq \mathcal{N}(L,\phi,F,\mathcal{X})(x) \triangleq F(x) - \vartheta^{*\tau}\phi(x)$. It is worth noticing that from property (P1) $\sup_{x\in\mathcal{X}} \nu(x)$ can be made arbitrarily small by appropriately selecting L. In general, $\sup_{x\in\mathcal{X}} \nu(x)$ becomes smaller whenever L increases.

II. AFT for Function Optimization

We will first introduce the AFT algorithm under the framework of stochastic approximation. More specifically, consider the problem of finding the parameter vector θ^* that minimizes the bounded-from-below objective function $f(\theta, x)$ in the case where the closed form of this function is not known and only noisy measurements of this function are available:

$$J = f(\theta, x) + \xi \tag{2.1}$$

Here ξ denotes the noise concatenating the measurement, $f(\cdot)$ is an at least C^2 bounded-from-below *unknown* function, θ is a multidimensional *control parameter vector* or *decision variables vector*, and x is a multidimensional vector of input variables. The problem at hand is to find the parameter vector θ^* that minimizes the average of f wrt x, i.e.,

$$\theta^* = \arg\min_\theta E\left[f(\theta, x)\right]$$

In many practical problems, the vector x may not be available for measurement, but it is natural to assume that an estimation $\bar{x}$ of x is available. Note that constraints on θ may be also imposed (e.g., θ could belong to a bounded set Θ, in which case the above minimization should take place for $\theta \in \Theta$). In the sequel we will assume for simplicity that θ can take

any value; the generalization for the case where $\theta \in \Theta$ is straightforward.

We are now ready to introduce the Adaptive Fine-Tuning (AFT) algorithm:

AFT Algorithm

Step 0: Choose $\theta^0, L_r, L_h, L_g, \alpha, \beta, \gamma$. Set $\ell = 0$.

Step 1: Set $\ell := \ell + 1$. Set

$$\theta^\ell = \theta^{\ell-1} + \alpha\Delta^{\ell-1} \tag{2.2}$$

where $\Delta^{\ell-1}$ is a vector of i.i.d. entries with zero mean and variance equal to 1. Calculate $J^\ell = J(\theta^\ell, x^\ell) + \xi^\ell$ (see comment C1).

$\vdots$

Step L_r: Set $\ell := \ell + 1$. Set

$$\theta^\ell = \theta^{\ell-1} + \alpha\Delta^{\ell-1} \tag{2.3}$$

where $\Delta^{\ell-1}$ is defined similarly to the previous step. Calculate $J^\ell = J(\theta^\ell, x^\ell) + \xi^\ell$(see comment C1).

Step $L_r + 1$: Set $\ell := \ell + 1$. Define the LUA with L_g regressor terms

$$\hat{J} = \vartheta^\tau \phi(\theta, \bar{x}^\ell) \tag{2.4}$$

where $\bar{x}^\ell$ denotes an estimate of the actual x^ℓ (see comments C2, C3).
Find the vector ϑ^m that minimizes the LUA least-square error for the last L_h iterations:

$$\vartheta^m = \arg\min_{\vartheta} \sum_{\lambda=max\{1,\ell-L_h\}}^{\ell-1} \left(J^\lambda - \vartheta^\tau \phi(\theta^\lambda, \bar{x}^\lambda)\right)^2 \tag{2.5}$$

(see comment C4)

Step $L_r + 2$: Calculate θ^ℓ according to

$$\theta_i^\ell = \theta_i^{\ell-L_r-1} - \beta \frac{\vartheta^{m\tau}\phi^+ - \vartheta^{m\tau}\phi^+}{2\gamma} \tag{2.6}$$

where $\phi^+ \triangleq \phi\left(\theta^{\ell-L_r-1} + \gamma e_i, \bar{x}^p\right)$, $\phi^- \triangleq \phi\left(\theta^{\ell-L_r-1} - \gamma e_i, \bar{x}^p\right)$, e_i is a vector whose entries are all zero except the i-th entry which is equal to 1, $\bar{x}^p$ is a predicted value of x^ℓ (see comment C5) and $\theta^{\ell-L_r-1}$ is the value of the parameter vector obtained in the previous algorithm iteration (see comment C6). Calculate $J^\ell = J(\theta^\ell, x^\ell) + \xi^\ell$(see comment C1).

Step $L_r + 3$: Adjust the values of $L_r, L_h, L_g, \alpha, \beta, \gamma$ if necessary (see comment C7). If a *termination condition* is not satisfied (see comment C8) Go to STEP 1; otherwise END.

Several comments are in order:

(C1) Note that Step 1 - Step $L_r + 3$ correspond to one full iteration of the AFT algorithm. Similarly to the SPSA algorithm, the parameter vector in each algorithm iteration is firstly randomly perturbed L_r times and the objective function is evaluated for these randomly perturbed parameter vector values. These values as well as past values of objective function evaluations are then used in order to approximate (estimate) the unknown objective function using a LUA. Note also that the SPSA algorithm can be regarded as a special case of the AFT algorithm. More specifically, in the case of the SPSA algorithm: (a) $L_h = L_r + 1$, that is, only the last $L_r + 1$ measurements are used for the estimation of the unknown objective function, while the AFT algorithm allows also for evaluations of previous iterations to be used for the approximation of the unknown objective function; (b) it can be easily seen that, if the regressor vector of the LUA used in (2.4) takes the form (here $i \in \{1, \ldots, L_r + 1\}$)

$$\phi_i(\theta, \bar{x}^\ell) = \begin{cases} 1 & \text{if } \left|\theta - \theta^{\ell-i-1}\right| \\ & \quad = \min_{j\in\{1,\ldots,L_r+1\}} \left|\theta - \theta^{\ell-j-1}\right| \\ 0 & \text{otherwise} \end{cases}$$

then, the vector ϑ^* that minimizes the LUA least-square error can be easily seen to be as follows:

$$\vartheta_i^* = J^{\ell-i-1}$$

With the above choices for the regressor vector and the vector ϑ^*, the AFT algorithm reduces to the SPSA algorithm. Of course, the above simple approximator does not possess the powerful approximation capabilities of other more elaborate LUAs.

(C2) In many cases, the measurement of the input vector x may be concatenated by measurement noise. Therefore it is natural to assume that there exists an estimate $\bar{x}^\ell$ of x which satisfies $\bar{x}^\ell = x^\ell + \zeta$, where ζ accounts for the measurement noise.

(C3) Contrary to other applications of LUA where the number of regressor terms L_g should be large enough to guarantee that the LUA can approximate complicated functions over large input sets, in the case of the proposed algorithm there is no need for a large number of regressor vectors. As it will be shown in the proof of the convergence properties of the proposed algorithm, it is sufficient if the LUA has enough regressor terms to efficiently approximate the unknown function $f(\cdot)$ in a small neighborhood around the most recent control parameter vector θ obtained at step $L_r + 2$. The size of this small neighborhood is proportional to the design parameters α, β, γ which should be kept small in order for the proposed algorithm to be efficient.

(C4) Many different algorithms can be used for the calculation of ϑ^m in (2.5). In our simulations a simple gradient algorithm was used [9] for this purpose. **(C5)** A possible choice for $\bar{x}^p$ is the average value of $\bar{x}^\ell$ or its value during the previous day. Of course, in the case where prediction models are available they can be used for the calculation of $\bar{x}^p$. In the application of the AFT algorithm to the fine-tuning of TUC's parameters, special attention has been paid to the calculation of $\bar{x}^p$ for reasons that are explained in section IV.

(C6) [Algorithm modification:] As in the case of the SPSA algorithm, the value of θ used in each algorithm iteration, is the most recent value obtained at the step $L_r + 2$. An intrinsic drawback of such a choice is that if during the last iteration a "bad" value for θ has been obtained then in the next iteration the algorithm should continue from this "bad" value. To overcome this problem the following value can be used in

(2.6) instead of the value $\theta^{\ell-L_r-1}$ obtained in the previous iteration

$$\theta^e = \arg \min_{\theta^\lambda, \lambda \in \{\ell-L_h,\ldots,\ell-1\}} J\left(\theta^\lambda, x^\lambda\right) \tag{2.7}$$

The value θ^e corresponds to the value of the control parameter vector θ that produced the best value for the objective function over the last L_h iterations. In our simulations, we used in (2.6), the value of θ^e as calculated in (2.7) instead of $\theta^{\ell-L_r-1}$ obtained in the previous iteration. Other choices for θ^e that take into account the "best" value under the worst demand scenario are also possible, e.g.,

$$\theta^e = \arg \min_{\theta^\lambda, \lambda \in \{\ell-L_h,\ldots,\ell-1\}} \left(J\left(\theta^\lambda, x^\lambda\right) \left|x^\lambda\right|^p\right),\ p>0 \tag{2.8}$$

(C7) The design parameters $L_r, \alpha, \beta, \gamma$ can be chosen as in the case of SPSA algorithm [16], [18]. For instance α, β, γ can be chosen to be positive slowly-decaying-to-zero sequences (more formally, zero-decaying, positive sequences with infinite ℓ_1 norms) while L_r can be chosen to take any positive value. However, the AFT algorithm - after experimenting with a variety of choices for $L_r, \alpha, \beta, \gamma$ - was found to perform more efficiently under a different choice for $L_r, \alpha, \beta, \gamma$, which is described in the simulation section.

(C8) Many different *termination conditions* for terminating the algorithm may be employed. For instance, the following termination condition can be used

$$J^\ell \leq min_{\lambda \in \{1,\ldots,\ell-1\}} J^\lambda + \eta$$

where η is a small positive design constant. Alternative termination conditions used in similar algorithms may be also employed.

The following theorem establishes that the proposed algorithm approximates the behaviour of the standard gradient-descent method. Note that contrary to [16], [18] where the convergence properties of the SPSA algorithm are established under a stochastic framework, our proof will follow a deterministic analysis; a stochastic framework may be, of course, followed as well.

Theorem 1: Assume that $\bar{x}^p$ in (2.6) is set equal to $\bar{x}^{\ell-L_r-1}$. Assume also that

(A1) The function $f(\cdot)$ is at least C^2. Let also $\bar{x}^\ell = x^\ell + \zeta$, where ζ denotes the input vector measurement noise.

(A2) The regressor vector $\phi(\cdot)$ satisfies the following persistency of excitation condition:

$$c_\alpha I \leq \sum_{\lambda=\ell-L_h+1}^{\ell} \phi(\theta, \bar{x}^\ell)\phi^\tau(\theta, \bar{x}^\ell)$$

where c_α is an $\mathcal{O}(\alpha)$ positive constant.

Then, the following holds:

$$\theta^\ell = \theta^{\ell-L_r-1} - \beta \frac{\partial f}{\partial \theta}(\theta^{\ell-L_r-1}, x^{\ell-L_r-1}) + \varepsilon\left(\alpha, \beta, \gamma, L_g, \epsilon^{\ell-1}\right) \tag{2.9}$$

where $\varepsilon\left(\alpha, \beta, \gamma, L_g, \epsilon^{\ell-1}\right) = \mathcal{O}(1/\alpha) + \frac{\mathcal{O}([\alpha,\beta,\gamma])}{\mathcal{O}(L_g)} + \mathcal{O}(\xi) + \mathcal{O}(\zeta) + \mathcal{O}(\epsilon^{\ell-1})$ and $\epsilon^{\ell-1}$ is an exponentially-zero-decaying nonnegative term.

Proof: Consider an instant at Steps L_r+1-L_r+2 of the AFT algorithm and let $\Theta_\ell^{\alpha,\beta,\gamma}$ denote the smallest closed subset satisfying

$$\theta^\lambda \in \Theta_\ell^{\alpha,\beta,\gamma},\ \forall \lambda \in \{\ell-L_h,\ldots,\ell-1\}$$

Obviously, the subset $\Theta_\ell^{\alpha,\beta,\gamma}$ depends on the choice of the design parameters α, β, γ; more precisely, the "smaller" these parameters are, the "smaller" the subset $\Theta_\ell^{\alpha,\beta,\gamma}$ is. Let us rewrite the gradient of $f(\cdot)$ as follows:

$$\frac{\partial f}{\partial \theta}(\theta, x) \quad = \quad \frac{\partial f}{\partial \theta}(\theta, \bar{x}) + f_n(\theta, x-\bar{x})$$

where because of assumption (A1) $\frac{\partial f}{\partial \theta}(\theta, x)$ is locally Lipschitz continuous and $f_n(\theta, x-\bar{x})$ is a function satisfying

$$|f_n(\theta, x-\bar{x})| \leq c_1 |x-\bar{x}|,\ \forall \theta \in \Theta_\ell^{\alpha,\beta,\gamma} \tag{2.10}$$

for some positive constant c_1. Let us fix the number L of the regressor terms of the LUA (2.4). By defining (see subsection I.D) $\vartheta^* \triangleq \mathcal{W}(L, \phi, f, \Theta_\ell^{\alpha,\beta,\gamma}) \triangleq \arg\min_\vartheta \sup_{\theta \in \Theta_\ell^{\alpha,\beta,\gamma}} |f(\theta, \bar{x}) - \vartheta^{*\tau}\phi(\bar{x})|$ and $\nu(\theta) \triangleq \mathcal{N}(L, \phi, f, \Theta_\ell^{\alpha,\beta,\gamma})(\theta) \triangleq f(\theta, \bar{x}) - \vartheta^{*\tau}\phi(\theta, \bar{x})$, we have

$$f(\theta, x) \quad = \quad \vartheta^{*\tau}\phi(\theta, \bar{x}) + \nu(\theta) + f_n(\theta, x-\bar{x})$$

therefore we have

$$f(\theta, x) - \hat{J} \quad = \quad \tilde{\vartheta}^\tau \phi(\theta, \bar{x}) + \nu(\theta) + f_n(\theta, x-\bar{x})$$

where $\tilde{\vartheta} \triangleq \vartheta^* - \vartheta^m$ denotes the parameter estimation error. Stantard results from the theory of parameter estimation (see e.g., [9], [11]) can be used to establish that, due to the persistency of excitation property (A2) the following is satisfied

$$\left|\tilde{\vartheta}^\tau\right| \leq \bar{c}_\alpha + c_2 |x-\bar{x}| + c_3 |\nu(\theta)| + c_3 |\xi| + \epsilon$$

where c_α is an $\mathcal{O}(1/\alpha)$ positive constant, c_2, c_3 are positive constants, ϵ is an exponentially-zero-decaying term and, thus, we finally obtain that

$$\begin{aligned} \frac{\partial f}{\partial \theta}(\theta, x) \quad &= \quad \frac{\partial \vartheta^{m\tau}\phi}{\partial \theta}(\theta, \bar{x}) + \frac{\partial \tilde{\vartheta}^\tau \phi}{\partial \theta}(\theta, \bar{x}) + \frac{\partial \nu}{\partial \theta}(\theta) \\ &\quad + \frac{\partial f_n}{\partial \theta}(\theta, x-\bar{x}) = \frac{\partial \vartheta^{m\tau}\phi}{\partial \theta}(\theta, \bar{x}) + \bar{\nu} \end{aligned}$$

where

$$\begin{aligned} \bar{\nu} \quad &= \quad \frac{\partial \tilde{\vartheta}^\tau \phi}{\partial \theta}(\theta, \bar{x}) + \frac{\partial \nu}{\partial \theta}(\theta) + \frac{\partial f_n}{\partial \theta}(\theta, x-\bar{x}) \Longrightarrow \\ |\bar{\nu}| \quad &\leq \quad (\bar{c}_\alpha + c_2 |x-\bar{x}| + c_3 |\nu(\theta)| + c_3 |\xi|)\, c_4 + c_5 |x-\bar{x}| \\ &\leq \quad \mathcal{O}(1/\alpha) + \mathcal{O}(\zeta) + \mathcal{O}(\xi) + c_6 |\nu(\theta)| + \mathcal{O}(\epsilon) \end{aligned}$$

where c_4 is a positive constant satisfying $c_4 = \sup_{\theta \in \Theta_\ell^{\alpha,\beta,\gamma}, \bar{x}} \left|\frac{\partial \phi}{\partial \theta}(\theta, \bar{x})\right|$, c_5 is a positive constant which depends on the size of $\Theta_\ell^{\alpha,\beta,\gamma}$ and $c_6 = c_3 c_4$. From standard results on approximation theory [8], the size of the modelling error term $|\nu(\theta)|$ is proportional to the size of the subset $\Theta_\ell^{\alpha,\beta,\gamma}$ and inversely proportional to the number L_g of regressor terms. Therefore, we finally obtain that

$$\bar{\nu} \leq \mathcal{O}(1/\alpha) + \mathcal{O}(\zeta) + \mathcal{O}(\xi) + \frac{\mathcal{O}([\alpha, \beta, \gamma])}{\mathcal{O}(L_g)} + \mathcal{O}(\epsilon)$$

which concludes the proof. ■

Some more comments are in order:

(C9) Theorem 1 simply states that the AFT algorithm is equivalent to the standard gradient descent optimization algorithm plus a "disturbance" term consisting of two parts: the "size" of the first part can be made arbitrarily small by choosing the design parameters α, β, γ sufficiently small and/or the number of regressor terms L_g sufficiently large; while the "size" of the second part is proportional to the size of the noise and prediction errors.

(C10) Obviously, in the case where θ^e is used instead of $\theta^{\ell-L_r-1}$ in (2.6) (where θ^e is calculated via (2.7) or (2.8); see comment C6), inequality (2.9) in Theorem 1 should be replaced by the following inequality

$$\theta^\ell = \theta^e - \beta \frac{\partial f}{\partial \theta}(\theta^e, x^{\ell-L_r-1}) + \varepsilon\left(\alpha, \beta, \gamma, L_g, \epsilon^{\ell-1}\right) \quad (2.11)$$

(C11) Note that, in order to keep the proof of Theorem 1 as simple as possible, we have defined the symbol $\mathcal{O}(\cdot)$ in a more general fashion than that of the frequently used "order of" symbol $O(\cdot)$ defined as follows: the function $f_\alpha(\cdot)$ parametrized by the positive constant α is said to be $O(c(\alpha))$ where $c(\cdot)$ a nonnegative function, if there exists a positive constant K such that for all α and all x, $|f_\alpha(x)| \leq Kc(\alpha)$.

(C12) The proof of Theorem 1 can be extended to cover the cases where the function $f(\cdot)$ is not necessarily C^2. The proof in this case is very lengthy and therefore omitted here.

(C13) Assumption (A2) about persistency of the regressor vector $\phi(\cdot)$ which is crucial for the proof of Theorem 1, is not automatically satisfied by the AFT algorithm for any choice of the LUA. However, if the LUA employed is a polynomial-in-θ approximator (as the one chosen in our simulations), it can be easily seen that the random perturbation of θ in Steps $1, \ldots, L_r$ is sufficient for the regressor vector to satisfy assumption (A2).

III. AFT for Non-linear Control over a Time-Horizon

The proposed algorithm can be directly applied for the fine-tuning of non-linear control systems. More precisely, consider a non-linear discrete-time control system

$$z_{k+1} = g(z_k, u_k, d_k),\ z_0 = z(0) \quad (3.1)$$

where z_k, u_k, d_k denote the vectors of system states, control inputs, and exogenous signals, respectively, k denotes the time-index, and $g(\cdot)$ is a - possibly unknown - sufficiently smooth non-linear vector function. Suppose that the following control law is applied to the system (3.1)

$$u_k = \varpi(\theta, z_k) \quad (3.2)$$

where $\varpi(\cdot)$ is a known vector function and θ is the vector of control parameters. The performance of the controller (3.2) is evaluated through the following objective function

$$\begin{aligned} J\left(\theta; z_0, D_K\right) &= \pi_K(z_K) + \sum_{k=0}^{K-1} \pi_k\left(z_k, u_k\right) \\ &= \pi_K(z_K) + \sum_{k=0}^{K-1} \pi_k\left(z_k, \varpi(\theta, z_k)\right) \end{aligned} \quad (3.3)$$

where π_k are known nonnegative functions, K denotes the time-horizon over which the control law (3.2) is applied and $D_K \stackrel{\triangle}{=} [d_0, d_1, \ldots, d_{K-1}]$ denotes the time-history of the exogenous signals.

By defining $x = \mathrm{vec}\,(z_0, D_K)$, the minimization of (3.3) wrt θ - and thus the problem of constructing an efficient fine-tuning algorithm for θ - reduces to the minimization of (2.1). Therefore, the proposed AFT algorithm and Theorem 1 are directly applicable to the problem of fine-tuning the control parameters θ so that (3.3) is minimized. It should be noticed that the application of the proposed AFT algorithm to a non-linear system of the form (3.1)-(3.3) does not guarantee that the closed-loop solutions for the different choices of θ are stable. In our application of the fine-tuning of the traffic control strategy TUC, this practically means that during the fine-tuning process there may be days where the closed-loop system behaviour is very poor which can lead to severe congestion problems, complaints from the drivers and the authorities, etc.

In the simulations we partly overcame this problem by the use of off-line testing of the control decisions under the new control parameter vector θ before evaluating on-line the control strategy with the new θ. In this off-line testing the control decisions using the new θ and data from previous days are compared to past control decisions that achieved an acceptable behaviour. If the comparison does not identify any major differences then the new θ is accepted for evaluation. Of course, the problem of guaranteeing closed-loop stability during the application of the AFT algorithm is an open one and of major importance. Our future research focuses on the resolution of this problem.

IV. Application to the Fine-Tuning of TUC

A. Brief Introduction to TUC

TUC (Traffic-responsive Urban Control) consists of four distinct interconnected control modules that allow for real-time control of:

- green times (split),
- cycle time,
- offsets, as well as
- for the provision of public transport priority.

These four control modules are complemented by a fifth data processing module. All control modules are based on feedback concepts of various types, which leads to TUC's computational simplicity as compared to model-based optimisation approaches, without actually sacrificing efficiency. The split control and data processing modules were the first to be developed (see [5] for details) while the other three control modules were developed at a later stage (see [6] for details). In this paper, we will concentrate on the fine-tuning of the TUC split control module parameters. The split control module is the most difficult to fine-tune since it consists of many control parameters (typically equal to the number of stages times the number of links). Of course, the proposed algorithm can be applied to the fine-tuning of the other control modules of TUC as well. It is worth noticing that since the cycle and offset control modules consist of a small number of control parameters (e.g., 3 design parameters for the cycle control

module) it is expected that the performance of the proposed algorithm to the fine-tuning of all control modules will be as efficient as in the case of fine-tuning the split control module alone.

We next briefly describe the Split Control module of TUC. This is a network-wide control module, i.e. all available measurements are used to calculate the green time of each stage via the multivariable regulator

$$g(k) = g_N - Lx(k-1) \tag{4.1}$$

where $k = 0, 1, 2, \ldots$ is the discrete time index with sample time period typically equal to the cycle time C; the vector $g(k)$ includes the green times of all stages in all junctions to be applied during the next cycle; g_N is a pre-specified vector of fixed green times (fixed plan) whose impact on the resulting control was found to be limited; the vector $x(k)$ comprises the numbers of vehicles in each network link during the last cycle, estimated by the data processing module; L is the control matrix (with dimensions number-of-stages/number-of-links) which results from an off-line applied software code based on the Linear-Quadratic (LQ) regulator design procedure; the traffic data required to calculate L are: saturation flows of links; average turning rates at junctions; maximum numbers of vehicles $x_{i,max}$ in links. The aim of (4.1) is to balance the relative space occupancies $x_i/x_{i,max}$ 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; to this end the regulator (4.1) may apply an inherent gating, i.e. reduce the green time of links that feed a saturating road, even if these links are two or more junctions away. The green times for the stages of each junction resulting from (4.1) will generally not add up to a cycle and may also violate minimum-green constraints; a suitably designed knap-sack optimisation modifies the green times so as to satisfy these constraints but keep the relative proportions of the green times as close to the ones of (4.1) as possible. In other words, the actual green times applied to the network, are not the $g(k)$ calculated in (4.1) but the $\bar{g}(k)$ obtained as a solution to the following minimization problem:

$$\begin{array}{l} \min \sum_{j\in I_\ell} \left(g_j(k) - \bar{g}_j(k)\right)^2 \\ \text{subject to} \\ \sum_{j\in I_\ell} \bar{g}_j(k) = C - L_\ell \\ g_{j,min} \le \bar{g}_j(k) \end{array} \tag{4.2}$$

where I_ℓ is the set of all stages in the ℓ-th junction, while L_ℓ is the total lost time (due to stage switchings) in the same junction.

The number of vehicles $x_i(k)$ for the i-th link are estimated via the following function:

$$x_i(k) = x_{i,max} \frac{f_s(o_i(k), \ell_i)}{1 - b_i f_s(o_i(k), \ell_i)} \tag{4.3}$$

where $o_i(k)$ denotes the measured average time-occupancy (measured usually by loop-detectors located at a certain distance from the stop light) during the last cycle time; $f_s(\cdot)$ is an empirical function [13], [6] constructed from practical investigations; ℓ_i denotes the distance of the loop-detector from the stop line divided by the total link length; finally, the nonnegative "gating" constant b_i is used to increase the influence of particular links to TUC's control decisions. Typically, b_i is set equal to zero.

The TUC control law (4.1) is based on the linearized store-and-forward model [5]

$$x(k+1) = x(k) + \bar{B}g(k) \tag{4.4}$$

where $\bar{B}$ is a constant matrix depending on the saturation flows of links, the average turning rates at junctions and the cycle time. The control matrix L in (4.1) is constructed using the Linear-Quadratic (LQ) optimal control technique on the model (4.4); more precisely, L is calculated so as to minimize the following function:

$$J = \sum_{k=1}^{\infty} \left(x^\tau(k) Q x(k) + g^\tau(k) R g(k)\right) \tag{4.5}$$

where $Q = \mathrm{diag}\left(1/x_{i,max}\right), R = rI$, with r a positive design constant. The aim of the function J is to balance the relative space occupancies $x_i/x_{i,max}$ 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

Although the linear system (4.4) represents a simplified model of the highly non-linear and complicated traffic dynamics, the use of LQ optimal control techniques as described above for the development of the split control module of TUC, provides a computationally simple yet efficient technique for split control as proved in the various TUC implementations [13].

B. The Traffic Network of Chania

Figure 1 displays the Chania urban traffic network, a typical urban traffic network containing all possible varieties of complex junction staging. The junctions are represented by nodes (junctions with common signalling have the same number, e.g. 1A, 1B, 1C) and the links are represented by arrows. Each network link corresponds to a particular junction phase. As it is seen in Figure 1 there are cases where there are more than one links (phases) from a particular junction to another (e.g., links L11 and L12 from junction 5 to junction 7); note that in the case of the network used in [17] there is only one link (stage) from a particular junction to another. Typical detector locations within the Chania network links are either around the middle of the link or some 40 m upstream of the stop line. As in most traffic networks there are two traffic demand peaks daily (noon and evening peaks). A typical time-history of the network mean-speed is shown in Figure 2 (the mean speed time-history plotted in Figure 2 corresponds to the case where TUC split control is applied with TUC's L matrix equal to the matrix $L_{(7.53)}$ described below).

The macroscopic simulation tool METACOR [7] was used for the simulation experiments. The traffic network characteristics (saturation flows, turning rates) used in METACOR were suggested by the operators of the Traffic Control Centre (TCC) of the city; the fixed plan g_N in (4.1) was also provided by the operators (it is one of the 6 fixed plans used by the TCC). A *basic traffic demand scenario* (time-history of vehicles entering the network in the network origins during the day) was designed based on actual measurements.

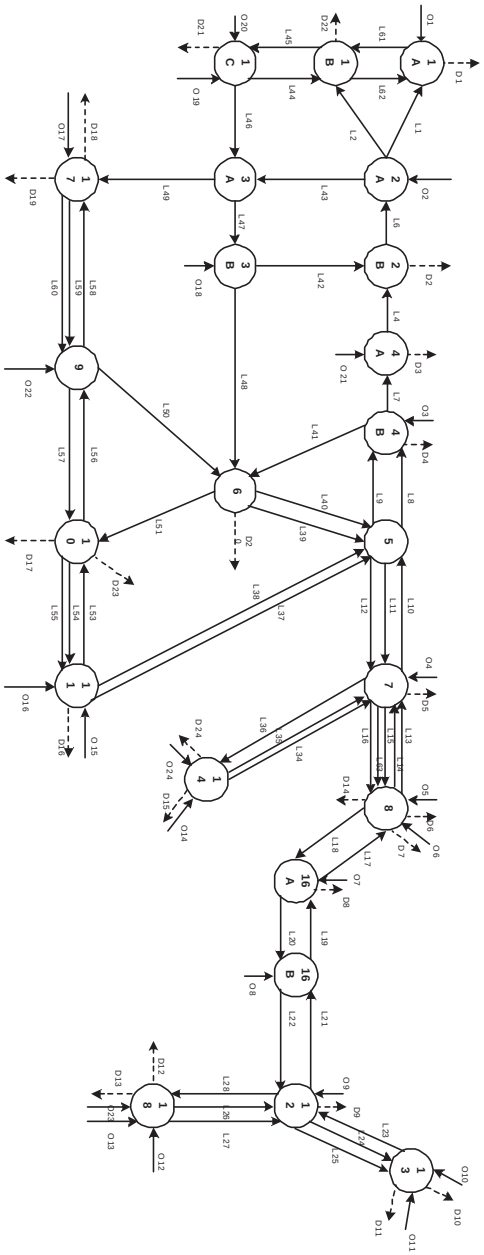

Fig. 1. The Chania urban traffic network.

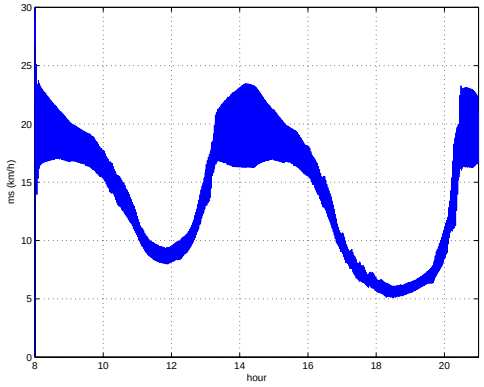

Fig. 2. Typical mean-speed daily time-history.

C. Choice of AFT Structure and Parameters

For the design of the LUA of the AFT algorithm for the particular application, a careful choice for the vectors $\bar{x}^{\ell}, \bar{x}^{p}$ had to be made. Note that in this particular application, the input vector x in (2.1) (or the vector $D_K \stackrel{\triangle}{=} [d_0, d_1, \ldots, d_{K-1}]$ in (3.3)) denotes the time-history of the number of vehicles entering the network in each of the network origins. Given the fact that there are a total of 22 origins in the network and a daily demand scenario corresponds to 14 hours, it would be computationally cumbersome to use vectors $\bar{x}^{\ell}, \bar{x}^{p}$ whose entries correspond to estimates of the vector x at small time-intervals (e.g., if the entries of $\bar{x}^{\ell}, \bar{x}^{p}$ correspond to 5-minute estimates, the total size of these vectors is equal to 3696 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. In our experiments, we found that it suffices to use an estimate every 3.5 hours (i.e., four different estimates for the demand of the whole day); this estimate was calculated by taking the average number of vehicles entering a particular origin over the 3.5 hours period and resulted in vectors $\bar{x}^{\ell}, \bar{x}^{p}$ of dimension equal to 88. A second arising problem was due to the calculation of the average demand based on flow measurements provided by loop-detectors; these measurements correspond to the actual demand when the queue at the origin link is small, but may severely underestimate the demand when the queue becomes long enough to cover the loop detector. In other words, when there is high demand or the particular control strategy cannot efficiently manage the queues at the origins, it is most likely that the flow measurements underestimate the actual demand. To partly overcome this problem, the vector $\bar{x}^{p}$ used in the AFT algorithm was the average over 3.5 hours flow measurement at each origin that corresponded to the best daily performance (in mean speed) obtained so far. In this way, we avoided the risk of using a $\bar{x}^{p}$ that corresponds to a non-efficient choice for the control parameter vector; of course, the vector $\bar{x}^{\ell}$ in Step $L_r + 1$ was the average over 3.5 hours flow measurement at each origin of the most recent daily experiment.

The LUA chosen for the algorithm implementation is a third-order - in θ and $\bar{x}^{\ell}$ - polynomial approximator taking the form:

$$\hat{J} = \vartheta_0 + \sum_{i=1}^{N} \vartheta_i \theta_j \bar{x}_k^{\ell} + \sum_{i=1}^{N} \vartheta_{i+N} \theta_h \theta_j \bar{x}_k^{\ell} \qquad (4.6)$$

where $\hat{J} \equiv \hat{mis}$ denotes the estimate of the actual network mean speed, and ϑ_i are the LUA constant parameters estimated through minimization of (2.4). The indices j, k, h were randomly chosen in each algorithm iteration; in other words, instead of taking all the possible combinations in the above third-order polynomial approximator, only a limited number of randomly chosen combinations was considered at each algorithm iteration. After experimenting with different values for N, we concluded that a value $N = 300$ was sufficient for an efficient AFT algorithm implementation. Note that with the above choice for the LUA, it can be easily seen that the

random perturbation Steps $1 - L_r$ of the AFT algorithm are sufficient for the persistency of excitation assumption (A2) to always hold. Finally, since our aim is to maximize the performance measured in network mean speed, θ^ℓ in Step $L_r + 2$ is calculated by using the following equation (see comment (C6))

$$\theta_i^\ell = \theta_i^e + \beta \frac{\vartheta^{m\tau}\phi^+ - \vartheta^{m\tau}\phi^-}{2\gamma} \tag{4.7}$$

The AFT design parameters were chosen as follows:

- L_r was chosen initially equal to 5 and was set equal to 1 after the first algorithm iteration. In other words, for the first 5 days of the experiments, random perturbations around the initial control parameter vector were used for each day, while after that, a random perturbation around the current control parameter vector was used every second day. The random perturbations for the first 5 days were used in order to provide the LUA with "enough" information about the unknown function $f(\cdot)$, while the choice of $L_r = 1$ for the rest of the experiments was chosen in order to minimize the instability behaviour that a random perturbation of θ may cause.
- The parameter L_h was set equal to 30, that is, the LUA approximator was using the 30 most recent experiments to obtain an approximation of the unknown function $f(\cdot)$. Since - as already mentioned in comment (C3) - the purpose of the LUA is to approximate the unknown function $f(\cdot)$ around a small neighborhood of the most recent control parameter vector, it does not make sense to set L_h large. In the case where L_h is chosen to be large, information from outside this small neighborhood will be introduced to the LUA, which may result in reducing its approximation capabilities.
- The parameters α and γ were kept constant and equal to small positive values (e.g., $\alpha = 0.01$ and $\gamma = 0.1$). It is worth noticing that different choices for the parameters α and γ (e.g. $\alpha, \gamma \in [0.001, 0.1]$) produced similar performance results; in other words, the algorithm seems to be robust with respect to the choice of the parameters α and γ. For the case of the gradient descent step-size β, the following simple rule was used: β was initially set equal to 0.01. After each algorithm iteration, if the measured performance was better than this of the previous iteration (i.e., if $J^\ell > J^{\ell - L_r - 1}$) then the value of β was increased according to $\beta = \beta * 5$. The increasing of β continued as long as $J^\ell > J^{\ell - L_r - 1}$; if $J^\ell \leq J^{\ell - L_r - 1}$ then β was reset to 0.01.
- η (defined comment (C8)) was set equal to 0.01.

D. Simulation Experiments

In all simulation experiments, a constant cycle time of 90 seconds was selected, and the offsets (time-differences of the start of each cycle) between adjacent junctions were constant, too. The following definition will help us explain the set up of the different experiments performed:

Definition 1: In the following, we will say that a value x_p is a randomly perturbed $m\%$-width version of a nominal value x, if the randomly perturbed value is calculated according to

$$x_p = x + \frac{m}{100} r$$

where r is a zero-mean uniform random value in $[-1, 1]$. In other words, if we say that a demand scenario is a randomly perturbed $m\%$-width version of the basic scenario, we will mean that each element of the randomly perturbed scenario was calculated according to the above formula with $m = 40$ and x_p the corresponding element of the basic scenario.

Different L matrices were designed based on assumed knowledge of the traffic network characteristics. In order to distinguish the different L matrices, we will use the notation $L_{(ms)}$ where ms is a number (in km/h) that corresponds to the average mean speed of the whole network obtained for the basic demand scenario when TUC is using a particular L matrix. For instance, the matrix $L_{(7.53)}$ is the matrix whose corresponding average network mean speed is 7.53 km/h for the basic demand scenario. Four different L matrices were designed:

- The matrix $L_{(7.53)}$ which was designed by assuming perfect knowledge of the network characteristics (saturation flows, turning rates).
- The matrices $L_{(6.55)}, L_{(6.2)}$ which were calculated by using randomly perturbed 40%-width versions of the actual values of the network characteristics (i.e., the saturation flows and the turning rates used in the design of these L matrices were randomly perturbed 40%-width versions of the ones used in the METACOR simulator and in the design of the matrix $L_{(7.53)}$).
- The matrix $L_{(3.2)}$ whose entries are all equal to zero (i.e., zero knowledge of the network characteristics). Note that in this case, only the fixed plan g_N is applied (i.e., no real-time control is applied), and the resulting mean speed is much less than in the cases where TUC operates with an L matrix based on uncertain or perfect knowledge.

Two different Experiments Sets were conducted:

Experiment Set 1: In this experiment set, each day a randomly perturbed 5%-width version of the basic demand scenario was used. Note that a 5%-width random perturbation makes small difference in the whole network's traffic behaviour. In this experiment set we intended to investigate the capabilities of the proposed algorithm under different choice of the algorithm's design parameters and this was the reason for assuming small variation in daily demands. Within Experiment Set 1, three different fine-tuning approaches were tested:

1.a In this case, the 544 most important entries of the matrix L were fine-tuned. As a matter of fact the L matrix has a total of 2982 entries (equal to [number of stages$= 42$] $\times$ [number of links $= 71$]); most of these entries have relatively small values since they correspond to links quite far from the junction of a particular stage and hence it does not seem necessary to fine-tune all of the entries of the L matrix. By checking the entries of the matrices $L_{(7.53)}, L_{(6.55)}, L_{(6.2)}$ we concluded that only 544 of these entries correspond to values that are not negligible. The simulation results of the proposed

fine-tuning approach for this case and for different initial L matrices and different choices of the algorithm parameters α, γ are summarized in Figures 3.a and 3.b. In all of the experiments of Figures 3.a and 3.b the gating parameters b_i were equal to 0.

1.b In this case, the matrix L was kept constant while the gating parameters b_i were fine-tuned using the proposed approach. Initially, the gating parameters were set equal to 0. Figure 3.c summarizes the simulation results for different L matrices. Note that one of the L matrices used (namely matrix $L_{(8.1)}$) was obtained after the fine-tuning of the entries of the L matrix (shown in Figure 3.a).

1.c As in the previous case, the matrix L was kept constant. The TUC control (4.1) was modified to

$$g(k) = g_N - Lx(k-1) + NN(k-1) \qquad (4.8)$$

where $NN(k-1)$ was the output of a multilayer neural network. Two different neural networks were used, one receiving as inputs the measurements $x(k-1)$ and the second receiving as inputs the measured flows $f(k-1)$ at the detector locations over the previous cycle. The initial neural network weights were chosen equal to zero. Figure 3.d summarizes the simulation results for different choices of the L matrix. As in case 1.a, the gating parameters b_i were equal to 0.

Experiment Set 2: In this experiment set, the basic demand scenario was randomly perturbed to an average 50% of its values in order to create 9 more basic demand scenarios; each day a random perturbation (with average 5% of the nominal values) of one of these 10 basic scenarios[1] was used. In this way, the efficiency of the proposed approach to demand scenarios with high variation was examined. Within the Experiment Set 2 only the fine-tuning of the 544 most important entries of the matrix L was tested. Figure 4 summarizes the simulation results for two different choices of the initial L matrix. In both cases the values of the gating parameters b_i were equal to 0.

An important issue raised in the Experiment Set 1 was the problem occurred in some of the simulation runs where a particular choice for θ^ℓ leads to a poor behaviour, especially in the case of experiment set 1.a with initial L matrix equal to $L_{(3.2)}$ (see Figure 3.b), where at certain days the mean speed achieved reaches very small values. To partly overcome this stability problem, in Experiment Set 2 we modified the AFT algorithm in Steps $1, \ldots, L_r$ as follows: at each of these steps the randomly perturbed control parameter value obtained in (2.2)-(2.3) is firstly *off-line* checked whereby the previous-day occupancy measurements are used to calculate the control decisions $\bar{g}(k)$ based on the new perturbed control parameter vector; if these decisions are significantly different from the ones obtained *on-line* on the previous day, then this perturbed control parameter vector is rejected and a new one is obtained by use of (2.2)-(2.3). This procedure is repeated until a perturbed control parameter vector is obtained whose off-line control decisions are not significantly different from the previous day's. The comparison was taking place by checking the norm of all the differences between the on-line and off-line control decisions; if the norm exceeded a certain threshold, then the particular perturbed control parameter vector is rejected.

[1]The same sequence of scenarios was used throughout this experiment set; this was done in order to make easier the performance of the AFT algorithm by comparing the performance at the current iteration with the one 10 iterations back.

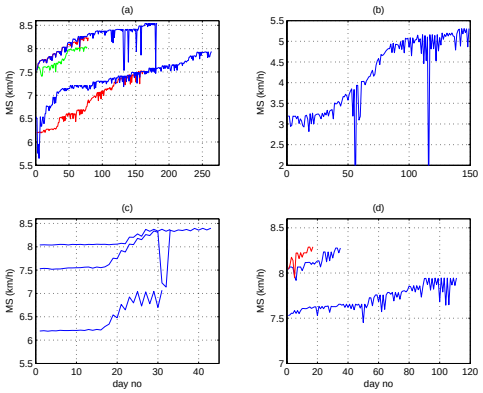

Fig. 3. Mean-speed results for the case of Experiment Set 1: (a) Experiment Set 1.a for different choices of the initial L matrix ($L = L_{(7.53)}, L_{(6.55)}, L_{(6.2)}$) and different choices for the design parameters α and γ, (b) Experiment Set 1.a with initial L matrix equal to $L_{(3.2)} = 0$, (c) Experiment Set 1.b for different choices of the L matrix ($L = L_{(8.1)}, L_{(7.53)}, L_{(6.2)}$), (d) Experiment Set 1.c. where the blue curves correspond to the controller $g(k) = g_N - Lx(k-1) + NN(x(k-1))$ for different choices of the matrix L ($L = L_{(8.1)}, L_{(7.53)}$), and the red curve corresponds to the controller $g(k) = g_N - Lx(k-1) + NN(f(k-1))$ with $L = L_{(8.1)}$.

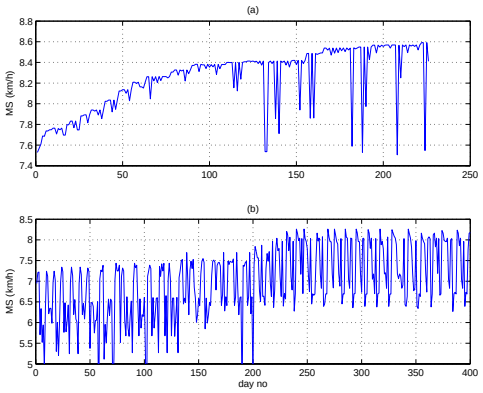

Fig. 4. Mean-speed results for the case of Experiment Set 2: (a) initial L matrix equal to $L_{(7.53)}$; (b) initial L matrix equal to $L_{(6.55)}$.

Several conclusions are drawn by inspecting Figures 3, 4:

- The AFT algorithm can very effectively fine-tune the control parameters of the system. We repeat at this point that application of the SPSA algorithm to the same fine-tuning problem described in this section failed to produce satisfactory results (in most of the cases, no improvement was observed during the application of the SPSA algorithm). Both for low variation of demand (experiment set 1 with 5%-width demand variation) and for high variation of demand (experiment set 2 with 40%-width demand variation), the fine-tuning of the entries of TUC's L-matrix using the AFT algorithm resulted in

an average improvement from 7.5 km/h to 8.5 km/h (or 13.3% improvement) in the case of perfect knowledge of the network characteristics, from 6.4 km/h - 7.9 km/h (or 23.4% improvement) in the case where the uncertainty on the network characteristics is random with 40%-width, and from 3 km/h to 5.2 km/h (or 73.3% improvement) in the case where both TUC and the AFT algorithm assume zero knowledge about the network characteristics.

- The TUC design principles lead to an efficient traffic controller with a satisfactory behaviour even without fine-tuning. In fact the controller that is based on perfect knowledge of network characteristics (i.e., the approach that corresponds to $L_{(7.53)}$), leads to a traffic performance which is not much lower than the one obtained after fine-tuning; moreover, the fine-tuning process that is based on a poor TUC design (i.e., the approach that corresponds to $L_{(6.55)}$) leads to a performance that is only slightly better than the one achieved by the matrix $L_{(7.53)}$. In a nutshell, TUC split control leads to a quite satisfactory behaviour, which, of course, can be improved further by use of the AFT algorithm.
- Fine-tuning of the gating parameters b_i can lead to a faster performance improvement when compared to fine-tuning of the entries of TUC's L-matrix (compare Figure 3.c with Figure 3.a where similar improvement is achieved in less days for the case of fine-tuning of b_i's). However, the fine-tuning of b_i's cannot achieve as much improvement as in the case of fine-tuning of the entries of TUC's L-matrix. Therefore, the best fine-tuning policy could be to firstly fine-tune the entries of TUC's L-matrix until performance is not improved anymore and then to fine-tune the gating parameters b_i.
- The use of neural network control $NN(k-1)$ additionally to the TUC linear split controller in (4.8) resulted in worse fine-tuning results than those of the linear controller (4.1). This in conjunction with the fact that the multilayer neural network used to represent $NN(k-1)$ can virtually approximate the behaviour of any non-linear controller, leads us to the conclusion that replacement of the linear controller (4.1) by a non-linear controller will not improve the capabilities of the Split Control module of TUC.
- From the simulation results it is apparent that there are many different local minima for the matrix L (see e.g., Figure 3.a where different initial L matrices converge to L matrices with different performance; moreover, in the case of initial L matrix equal to $L_{(7.53)}$ the same initial matrix but different AFT design parameters lead to different final L matrices); however, in all cases the final L matrices where AFT converges lead to a satisfactory behaviour.
- As mentioned in subsection IV.C, due to the use of loop-detectors, the demand may be severely underestimated when the queue becomes long enough to cover the loop detector, and, as a result, the estimate $\bar{x}^{\ell}, \bar{x}^{p}$ may be very noisy in many cases. It is therefore expected that the performance of the proposed algorithm will be improved if instead of loop-detector, image sensors - which can more accurately measure the link queues - are used for the measurement of traffic flow/queues within traffic nework links. Of course, image sensors are rarely used by urban traffic control systems mainly due to their high cost.

V. Conclusions

In this paper a new algorithm for fine-tuning of parameters of non-linear control systems has been proposed and analyzed both by mathematical analysis and by means of simulations of the traffic control strategy TUC as applied to the traffic network of the city of Chania. The proposed approach is based on a concept similar to the Simultaneous Perturbation Stochastic Approximation (SPSA) algorithm, which has been applied and analyzed for the fine-tuning of a simple traffic control system. The difference between the SPSA algorithm and the proposed approach is that, while SPSA uses an approximation of the gradient (of an appropriate objective function) that uses only the most recent fine-tuning experiments, in the proposed approach the approximation of the gradient function is performed by using a linear-in-the-parameters approximator that incorporates information of a user-specified time-window of past experiments. Mathematical analysis of the proposed approach establishes its convergence properties while SPSA can be regarded as a special case of the proposed approach.

Simulation results using the complicated traffic network of the city of Chania, establish the efficient performance of the proposed approach. It is worth noticing that the SPSA algorithm fails when applied to the fine-tuning of the TUC control parameters for the Chania traffic network. Finally, some useful conclusions about the efficiency of the TUC strategy are drawn.

References

[1] A.R. Barron, "Universal approximation bounds for superpositions of a sigmoidal function," *IEEE Transactions on Information Theory*, vol. 39, no. 3, pp. 930-945, 1993.

[2] Ch. Bielefeldt, C. Diakaki, and M. Papageorgiou, "TUC and the SMART NETS project," *Proc. IEEE 4th International Conference on Intelligent Transportation Systems*, Oakland, California, pp. 55-60, 25 - 29 August 2001.

[3] N.E. Cotter, "The Stone-Weierstrass theorem and its application to neural networks," *IEEE Trans. on Neural Networks*, vol. 1, no. 4, pp. 290-295, 1990.

[4] G. Cybenko, "Approximations by superpositions of a sigmoidal function," *Mathematics of Control, Signals, and Systems*, vol. 2, pp. 303-314, 1989.

[5] 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.

[6] C. Diakaki,V. Dinopoulou, K. Aboudolas, M. Papageorgiou, E. Ben-Shabat, E. Seider, and A. Leibov, "Extensions and new applications of the traffic signal control strategy TUC," *Transportation Research Record* no. 1856, pp. 202-216, 2003.

[7] N. Elloumi, H. Haj-Salem, and M. Papageorgiou, "METACOR: A macroscopic modeling tool for urban corridors," *TRISTAN II (Triennal Symposium on Transportation Analysis)*, Capri, Italy, Vol.1, pp. 135-150, June 23-28, 1994.

[8] J. A. Farrell and M.M.. Polycarpou. *Adaptive Approximation Based Control : Unifying Neural, Fuzzy and Traditional Adaptive Approximation Approaches*. Wiley Publishers, 2006.

[9] P. A. Ioannou and J. Sun, *Stable and Robust Adaptive Control*, Englewood Cliffs, NJ: Prentice Hall, 1995.

[10] E.B. Kosmatopoulos and M.A. Christodoulou, "Filtering, prediction, & learning properties of ECE neural networks," *IEEE Transactions on Systems, Man, and Cybernetics*, vol. 24, no. 7, pp. 971-981, July 1994.

[11] E. B. Kosmatopoulos, M. M. Polycarpou, M. A. Christodoulou, and P. A. Ioannou, "High-order neural network structures for identification of dynamical systems," *IEEE Transactions on Neural Networks*, vol. 6, no. 2, pp. 422-431, March 1995.
[12] E. B. Kosmatopoulos and M. A. Christodoulou, "Structural properties of gradient recurrent high-order neural networks," *IEEE Transactions on Circuits And Systems—II: Analog and Digital Signal Processing*, vol. 42, no. 9, September 1995.
[13] E. B. Kosmatopoulos, M. Papageorgiou, V. Dinopoulou, Ch. Bielefeldt, R. Morris, J. Mueck, A. Richards, F. Weichenmeier, "International comparative field evaluation of a traffic signal control strategy in three cities," *Transportation Research A*, vol. 40, no. 5, pp. 399-413, 2006.
[14] M.M. Polycarpou and P.A. Ioannou, "Identification and control of non-linear systems using neural network models: design and stability analysis," *Tech. Rep. 91-09-01*, Univ. of Southern Cal., Los Angeles, September, 1991.
[15] R.M. Sanner and J.-J.E. Slotine, "Gaussian networks for direct adaptive control", *Proc. American Control Conf., ACC91*, pp. 2153-2159, 1991; also in *IEEE Transactions on Neural Networks*, vol. 3, no. 6, pp. 837-863, November 1992.
[16] J.C., Spall, "Multivariate Stochastic Approximation Using a Simultaneous Perturbation Gradient Approximation," *IEEE Transactions on Automatic Control*, vol. 37, pp. 332-341, 1992.
[17] J.C. Spall and D.C. Chin, "Traffic-Responsive Signal Timing for System-Wide Traffic Control," *Transportation Research C*, vol. 5, pp. 153-163, 1997.
[18] J.C., Spall, "Adaptive Stochastic Approximation by the Simultaneous Perturbation Method," *IEEE Transactions on Automatic Control*, vol. 45, pp. 1839-1853, 2000.
[19] http://www.jhuapl.edu/SPSA/

PLACE PHOTO HERE

Elias B. Kosmatopoulos received the Diploma, M.Sc. and Ph.D. degrees from the Technical University of Crete, Greece, in 1990, 1992, and 1995, respectively. He is currently an Assistant Professor with the Department of Production Engineering and Management, Technical University of Crete (TUC), Greece and Deputy Director of the Dynamic Systems and Simulation Laboratory at TUC. Prior to joining TUC, he was Research Assistant Professor with the Department of Electrical Engineering-Systems, University of Southern California (USC) and a Postdoctoral Fellow with the Department of Electrical & Computer Engineering, University of Victoria, B.C., Canada.

Dr. Kosmatopoulos' research interests are in the areas of neural networks, adaptive optimization and control and intelligent transportation systems. He is the author of over 25 journal papers. Among his theoretical contributions the most important are the analysis of approximation, stability and learning capabilities of Recurrent High Order Neural Networks and the development and analysis of a switching adaptive controller for unknown dynamical systems.

PLACE PHOTO HERE

Markos Papageorgiou was born in Thessaloniki, Greece, in 1953. He received the Diplom-Ingenieur and Doktor-Ingenieur (honors) degrees in Electrical Engineering from the Technical University of Munich, Germany, in 1976 and 1981, respectively. From 1976 to 1982 he was a Research and Teaching Assistant at the Control Engineering Chair, Technical University of Munich. He was a Free Associate with Dorsch Consult, Munich (1982-1988), and with Institute National de Recherche sur les Transports et leur Scurit (INRETS), Arcueil, France (1986-1988). From 1988 to 1994 he was a Professor of Automation at the Technical University of Munich. Since 1994 he has been a Professor at the Technical University of Crete, Chania, Greece. He was a Visiting Professor at the Politecnico di Milano, Italy (1982), at the Ecole Nationale des Ponts et Chausses, Paris (1985-1987), and at MIT, Cambridge (1997, 2000); and a Visiting Scholar at the University of Minnesota (1991, 1993), University of Southern California (1993) and the University of California, Berkeley (1993, 1997, 2000).

Dr. Papageorgiou is the author of the books Applications of Automatic Control Concepts to Traffic Flow Modeling and Control (Springer, 1983) and Optimierung (Oldenbourg, 1991; 1996), the editor of the Concise Encyclopedia of Traffic and Transportation Systems (Pergamon Press, 1991), and the author or co-author of over 200 technical papers. His research interests include automatic control and optimisation theory and applications to traffic and transportation systems, water systems and further areas. He is an Associate Editor of Transportation Research-Part C and Chairman of the IFAC Technical Committee on Transportation Systems. He is a member of the Technical Chamber of Greece (TEE) and a Fellow of IEEE. He received a DAAD scholarship (1971-1976), the 1983 Eugen-Hartmann award from the Union of German Engineers (VDI), and a Fulbright Lecturing/Research Award (1997).

PLACE PHOTO HERE

Antigoni Vakouli received her Diploma Degree from the Department of Production & Management Engineering, Technical University of Crete, Greece. She is currently an M.Sc. student at the same department. Her main interests are in intelligent transport systems, traffic control and adaptive control.

Anastasios Kouvelas received his Diploma Degree from the Department of Production & Management Engineering, Technical University of Crete, Greece. He is currently an M.Sc. student at the same department. His main interests are in control of urban traffic networks and intelligent transport systems.