On congestion pricing Stackelberg games in dynamic traffic networks

Neurosimulation-based solution approach

TRAIL Research School, Delft, October 2008

Authors
Kateřina Staňková, MSc., Prof. dr. Geert Jan Olsder
Faculty of Electrical Engineering, Mathematics and Computer Science, Department of Mathematical Systems Theory, Delft University of Technology, the Netherlands
Dr. Michiel C.J. Bliemer
Faculty of Civil Engineering and Geosciences, Department of Transport & Planning, Delft University of Technology, the Netherlands

Contents

Abstract

In this paper the optimal toll design problem as a dynamic game of inverse Stackelberg type is investigated, with the road authority as a leader and drivers of the road network as followers. The road authority sets traffic-flow dependent tolls on some of the links so as to minimize the total travel time of the system, while the travelers choose their routes so as to minimize their perceived travel costs.

We formulate the dynamic optimal toll design problem with flow-dependent second-best tolling, discuss its properties, and present a numerical solution of the problem. The neurosimulator FAUN (Fast Approximation with Universal Neural Networks) is used to solve the minimizing problem. The algorithm performance is shown on small case studies.

In our case studies the flow-dependent tolling improved the system performance remarkably even with a rather simple toll function. This suggests that the traffic-flow dependent toll is a very promising tool for real applications. For the real-time applications, the parallelization of the solution method is a must.

Keywords

Road pricing, dynamic optimal toll design problem, logit-based stochastic user equilibrium, dynamic traffic assignment, first-best tolling, second-best tolling, Stackelberg games, inverse Stackelberg games, neural networks, FAUN.

1 Introduction & literature overview

Traffic congestion has become a big problem, especially in heavily populated metropolitan areas. With increasing occupancy of the road networks, the problem of congestion becomes more and more actual. Common methods that are used to alleviate congestion may be too expensive, difficult to apply, and not very efficient. When it is not easy to apply the standard methods, traffic congestion can be reduced by imposing appropriate tolls using a road pricing scheme.

The idea of reducing congestion via appropriate tolls led to the introduction of the so-called *optimal toll design problem* (Patriksson & Rockefellar (2002); Joksimovič et al. (2004)). This optimal toll design problem is a problem of the Stackelberg type (Başar & Olsder (1999); Bagchi (1984)), applied to the traffic environment with a road authority as a leader and travelers as followers. The aim of the road authority is to minimize its objective function, which is dependent on the travelers' decisions, by choosing optimal tolls for a subset of links (so-called tollable links), while the travelers minimize their individual travel costs. Their behavior is usually modeled by applying a *traffic assignment* (Patriksson (1994); Fisk (1980)).

There are extensive studies focusing on the *static* optimal toll design problem, i.e., on problems in which decisions of the players (travelers and the road authority) do not evolve in time (Verhoef (2002); Patriksson & Rockefellar (2002)). Although static models are still widely used, the theory and practice of *dynamic* models have evolved significantly over the last ten years. In the dynamic version of the optimal toll design problem the *dynamic traffic assignment* (DTA) applies (Bliemer (2001a)). DTA models typically describe route choice behavior of travelers on a transportation network and the way in which traffic dynamically propagates through the network.

If the travelers are assumed to have perfect information, the *deterministic user equilibrium* (DUE) applies (Wardrop (1952)), in both dynamic and static optimal toll design. Similarly, with imperfect information, the *stochastic user equilibrium* (SUE) applies, for example as a *logit-based stochastic equilibrium* (LB-SUE), see (Lotito et al. (2005)).

Considering possible tolling strategies there are two main research streams differing in the definition of the set of tollable links. With so-called first-best tolling (or pricing) all the links in the network can be tolled (Patriksson & Rockefellar (2002); Yilidirim & Hearn (2005)), with so-called second-best tolling not all links are tollable (Verhoef (2002)). The latter concept is clearly more applicable in practice.

Following extensive case studies of two-route congestion problems in static networks (Vickrey (1969); Defermos & Sparrow (1971); E. T. Verhoef & Rietveld (1996)) we have introduced its second-best variant, where the link tolls are functions of link and route flows in the network, for only a proper subset of all links. This fits within a theoretical framework of "inverse Stackelberg problems" (Olsder (2005); Staňková et al. (2006)).

This paper introduces an extension of our recent research to dynamic problems with SUE. Although some authors (e.g., Braid (1989); Arnott et al. (1990)) consider the step-wise second-best tolling, to the best of our knowledge no research dealing with the optimal toll design problem with the second-best tolling, the travelers driven by LB-SUE, and the aim to find optimal toll defined as a function of the traffic flows in the network has been done before. Since the problem is NP-hard, advanced optimization

techniques, which can be parallelized, should be used in order to speed up the solution process. In this paper an algorithm using neural networks is introduced as such an optimization technique. Neurosimulation is usually based on complete software emulation, i.e., inputs, outputs, neurons, synapses and weights are implemented in software. The FAUN (Fast Approximation with Universal Neural networks) neurosimulator enables supervised learning with artificial neural networks (ANN). A well trained ANN is a mathematical function which approximates the output of the input-output sample patterns reasonably (generalization). Here the word "unreasonable" summarizes different quality factors for the trained ANN, i.e., the mathematical function. The neurosimulator FAUN has already been employed to solve other problems in the domain of dynamic games, see, e.g., Breitner (2000); Mettenheim & Breitner (2005, 2006).

The contributions of the paper can be listed as follows:

- An algorithm solving the dynamic optimal toll design problem (with traffic-flow dependent toll) using neural networks is introduced and presented on a small case study. This algorithm is applicable on general networks.
- This problem is also a new application of the neurosimulator FAUN.
- We show that the flow-dependent tolling can never be worse than the flow-independent one.
- Paradoxical phenomena of optimal toll decreasing with actual traffic flows are presented and discussed.

This paper is organized as follows: In Section 2 the optimal toll design problem is presented, including some properties and examples of interesting phenomena, which will be explained analytically. In Section 3 neurosimulation and the neurosimulator FAUN 1.1 used for solving our problem are introduced. In Section 4 the algorithm, that is used for solving the problem, is presented. In Section 5 we show the case studies on a Chen network. Conclusions and future research are discussed in Section 6.

2 The optimal toll design problem

In this section the dynamic optimal toll design problem is introduced. This problem belongs to the class of so-called inverse Stackelberg games (Staňková et al. (2006), Staňková et al. (2008b)).

2.1 Preliminaries

Let $G(\mathcal{N}, \mathcal{A})$ be a road network defined by finite nonempty set of nodes $\mathcal{N}$ and finite nonempty set of links $\mathcal{A}$. Let $\mathcal{R} \subset \mathcal{N}$ and $\mathcal{S} \subset \mathcal{N}$ be finite nonempty sets of origin nodes (origins) and destination nodes (destinations), respectively, and let $\mathcal{RS} \subset \mathcal{N} \times \mathcal{N}$ be a subset of origin-destination pairs. For each origin-destination pair $(r, s) \in \mathcal{RS}$, where r is the origin and s is the destination, there is a travel demand $d^{(r,s),k} \in \mathbb{R}_+ \, [veh]$ on the number of travelers departing during k-th time interval from origin r to destination s. The network is assumed to be strongly connected, i.e., at least one route connects each (r, s)-pair. For each directed arc $a \in \mathcal{A}$ the following parameters are initially

given: link length s_a [km], maximum speed $\vartheta_a^{\max}$ [km/h], minimum speed $\vartheta_a^{\min}$ [km/h], critical speed ϑ_a^{crit} [km/h], jam density J_a^{jam} [pcu[1]/km], and the unrestricted link capacity C_a [pcu/h]. Dynamic link travel time for an individual user entering link a during k-th time interval ($k \in \mathcal{K}$) is defined as

$$\tau_a^k = \frac{s_a}{\vartheta_a^k}, \tag{1}$$

where the link speed ϑ_a^k [km/h] can be computed using *Smulders speed-density function* (see Smulders (1988)):

$$\vartheta_a^k = \begin{cases} \vartheta_a^{\max} + \frac{\vartheta_a^{crit} - \vartheta_a^{\max}}{J_a^{crit}} J_a^k, & \text{if} \quad J_a^k \le J_a^{crit}, \\ J_a^{jam} + \left(\vartheta_a^{crit} - \vartheta_a^{\min}\right) \frac{\frac{1}{J_a^k} - \frac{1}{J_a^{jam}}}{\frac{1}{J_a^{crit}} - \frac{1}{J_a^{jam}}}, & \text{if} \quad J_a^{crit} \le J_a^k \le J_a^{jam}, \\ \vartheta_a^k & \text{if} \quad J_a^k \ge J_a^{jam}, \end{cases} \tag{2}$$

with critical density J_a^{crit} [pcu/km] defined as $J_a^{crit} = \frac{C_a}{\vartheta_a^{crit}}$. Dynamic link cost as experienced by a single traveler entering link $a \in \mathcal{A}$ during k-th time interval is defined as

$$c_a^k = \alpha \tau_a^k + \theta_a^k, \tag{3}$$

where α is the traveler's value of time [€/h] and θ_a^k [€] is toll that a single traveler pays when entering link a during k-th time interval. The dynamic link travel times, tolls, and costs are additive, i.e.,

$$\tau_p^k = \sum_{a \in \mathcal{A}} \sum_{k' \in \mathcal{K}} \delta_{a,p}^{k,k'} \tau_a^{k'}, \quad \theta_p^k = \sum_{a \in \mathcal{A}} \sum_{k' \in \mathcal{K}} \delta_{a,p}^{k,k'} \theta_a^{k'}, \quad c_p^k = \sum_{a \in \mathcal{A}} \sum_{k' \in \mathcal{K}} \delta_{a,p}^{k,k'} c_a^{k'}. \tag{4}$$

Here $\delta_{a,p}^{k,k'}$ is a *dynamic route-link incidence indicator*, equal to one if drivers traveling over path $p \in \mathcal{P}$ and departing during k-th time interval $k \in \mathcal{K}$ reach link a during k'-th time interval and zero otherwise, and τ_p^k, θ_p^k, and c_p^k are dynamic route travel time, dynamic route toll, and dynamic route cost for travelers entering route $p \in \mathcal{P}$ during k-th time interval, respectively. Dynamic link flow rates are additive with respect to dynamic route flow rates, i.e.,

$$q_a^k = \sum_{a \in \mathcal{A}} \sum_{k' \in \mathcal{K}} \delta_{a,p}^{k,k'} f_p^{k'}, \tag{5}$$

where q_a^k [veh/h] is the dynamic link flow rate of drivers entering link a during k-th interval and $f_p^{k'}$ [veh/h] is the dynamic route flow rate of travelers entering route p during k'-th time interval. For all $(r,s) \in \mathcal{RS}$, $p \in \mathcal{P}^{(r,s)}$, and $k \in \mathcal{K}$, the route flow rates have to be *feasible*, i.e., vector of route flow rates $\mathbf{f}^{(r,s),k} \stackrel{\text{def}}{=} \left(f_1^{(r,s),k}, \ldots, f_{|\mathcal{P}^{(r,s)}|}^{(r,s),k}\right)'$ belongs to the set $Q^{(r,s),k}$ defined as

$$Q^{(r,s),k} \triangleq \left\{ \left(x_1, \ldots, x_{|\mathcal{P}^{(r,s)}|}\right)' : \textstyle\sum_{i \in \{1,\ldots,|\mathcal{P}^{(r,s)}|\}} x_i = d^{(r,s),k}, \quad x_i \ge 0, \quad \forall i \in \{1, \ldots, |\mathcal{P}^{(r,s)}|\} \right\}. \tag{6}$$

[1] passenger car units

From (5) and (6) it follows that also the link flow rates are feasible with respect to the route flow rates, i.e., vector of link flows $\mathbf{q}^{\mathcal{A},k} = \left(q_1^k, \ldots, q_{|\mathcal{A}|^k}\right)'$ belongs to set of *feasible link flows* $Q^{\mathcal{A},k}$ defined as

$$Q^{\mathcal{A},k} \triangleq \left\{ \left(y_1, \ldots, y_{|\mathcal{A}|}\right) : y_a = \sum_{a \in \{1,\ldots,|\mathcal{A}|\}} \sum_{k' \in \mathcal{K}} \delta_{a,p}^{k,k'} f_p^{k'}, \quad \forall a \in \mathcal{A} \right\}. \tag{7}$$

The link dynamics is defined by the *Dynamic Network Loading* (DNL) model. The DNL model is formulated as a system of equations expressing *link dynamics*, *flow conservation*, *flow propagation*, and *boundary constraints*. The DNL model is adopted from Chabini (2002) and is not further discussed here.

Drivers minimize their perceived travel costs. We assume that in equilibrium state no traveler can minimize her perceived travel costs by unilateral change of her route. So-called *dynamic logit-based stochastic user equilibrium* applies (see Bliemer (2001b); Lotito et al. (2005)).

2.2 Problem formulation

The problem to be solved is an inverse Stackelberg game with the road authority as the leader and the drivers as followers. The road authority sets tolls on so-called tollable links $\mathcal{T} \subset \mathcal{A}$ as mappings of traffic flows in the network so as to minimize the total travel time of the system. For each travel time interval the road authority imposes a vector of the link tolls $\Theta^k \stackrel{\text{def}}{=} \left(\theta_1^k(\cdot), \ldots, \theta_{|\mathcal{A}|}^k(\cdot)\right)$, $\theta_a^k(\cdot) : Q^{\mathcal{A},k} \rightarrow \mathbb{R}_0^+$, in such a way so as to minimize the total travel time of the system. Trivially, the problem of minimizing the total travel time

$$\sum_{k \in \mathcal{K}} \sum_{(r,s) \in \mathcal{RS}} \sum_{p^{(r,s)} \in \mathcal{P}^{(r,s)}} \tau_p^{(r,s),k} \cdot f_p^{(r,s),k}$$

is equivalent to minimizing congestion in the network.

Let matrix Θ be a matrix of toll functions set by the road authority on each link for each time interval, i.e.,

$$\Theta \stackrel{\text{def}}{=} \begin{pmatrix} \Theta^1(\cdot) \\ \vdots \\ \Theta^{|\mathcal{K}|}(\cdot) \end{pmatrix} = \begin{pmatrix} \theta_1^1(\cdot) & \ldots & \theta_{|\mathcal{A}|}^1(\cdot) \\ \vdots & \ddots & \vdots \\ \theta_1^{|\mathcal{K}|}(\cdot) & \ldots & \theta_{|\mathcal{A}|}^{|\mathcal{K}|}(\cdot) \end{pmatrix}, \tag{8}$$

where for each $k \in \mathcal{K}$ and $a \in \mathcal{A}$

$$\theta_a^k(\cdot) \begin{cases} = \mathbf{0}, & \text{if} \quad a \in \mathcal{A} \setminus \mathcal{T}, \\ \geq \mathbf{0}, & \text{otherwise.} \end{cases} \tag{9}$$

The main problem solved in this paper can be defined as follows:

(P) Find

$$\Theta^* = \arg\min_{\Theta} \sum_{k \in \mathcal{K}} \sum_{(r,s) \in \mathcal{RS}} \sum_{p \in \mathcal{P}^{(r,s)}} \tau_p^{(r,s),k} \cdot f_p^{(r,s),k}$$

subject to DNL, (1) - (5), LB-SUE, (9), while the link and route flows are feasible.

Remarks:

1. The total travel time is clearly nonlinear function of the link flows in the network. The problem (P) is a member of the class called nonlinear bilevel programming problems. In Bard (1991) it was shown that even the linear bilevel programming problems (with linear objective function) are NP-hard. Furthermore, in Hansen et al. (1992) it is shown that the linear bilevel programming problems are strongly NP-hard. From this it follows that the problem (P) is strongly NP-hard, too. The complete proof of this statement can be found in Staňková et al. (2008b). Since the problem (P) is strongly NP-hard, heuristic methods, like neurosimulation introduced in this paper, should be used to find a (satisfying) solution.

2. Please note that the problem of finding the uniform toll or time-varying toll is clearly a special case of (P). It follows from the observation that Stackelberg games are in subset of inverse Stackelberg games. For more information on this topic, see Staňková et al. (2008a), Başar & Olsder (1999).

3. The problem (P) has generally a nonunique solution (See Staňková et al. (2008b)) for proof of existence and nonuniqueness of a solution.

4. The solution to the problem (P) does not need to be a function increasing with the traffic flow rate, as illustrated on the following static[2] example with linear link travel time function. This phenomena will appear also in the case studies in Section 5.

   Example 2.1 (Toll decreasing with the traffic flow)**.** *Let us consider a simple example of a static optimal toll design problem on a three-link network with one origin-destination pair and travelers driven by deterministic user equilibrium (DUE). For the sake of simplicity we assume that both traffic demand for the origin-destination pair and value of time are utilized to one and the link cost and time functions are linear, i.e.,*

$$D = q_1 + q_2 + q_3, \tag{10}$$

$$c_1 = \alpha\tau_1 + \theta_1(q_1),\ c_2 = \alpha\tau_2 + \theta_2(q_2),\ c_3 = \alpha\theta_3, \tag{11}$$

$$\tau_1 = \beta_1 q_1 + \delta_1,\ \tau_2 = \beta_2\, q_2 + \delta_2,\ \tau_3 = \beta_3\, q_3 + \delta_3. \tag{12}$$

   with $D = 1$, $\alpha = 1$, $\beta_1 = 1$, $\beta_2 = 2$, $\beta_3 = 0.05$, $\delta_1 = 1.008$, $\delta_2 = 0.672$, $\delta_3 = 2$. *Then the total travel time* $F(q_1, q_2, q_3)$ *function can be computed as*

$$\begin{aligned} F(q_1, q_2, q_3) &= \sum_{j=1}^{3} q_{l_j}\, \tau_{l_j} \\ &= q_1\,(q_1 + 1.008) \\ &+ q_2(2\,q_2 + 0.672) + (1 - q_1 - q_2)(2.05 - 0.05\,q_1 - 0.05\,q_2) \\ &= 1.05\,q_1^2 + 2.05\,q_2^2 - 1.092\,q_1 - 1.428\,q_2 + 2.05 + 0.1\,q_1\,q_2. \end{aligned}$$

   Global minimum of $F(q_1, q_2, q_3)$ *is in* $q_1^* = 0.504$ *[veh/h],* $q_2^* = 0.336$ *[veh/h],* $q_3^* = 0.16$ *[veh/h] and* $F(q_1^*, q_2^*, q_3^*) = 1.534912$ *[h]. This is the best what the road authority can obtain.*

[2] with $\mathcal{K} \stackrel{\text{def}}{=} 1$

Let us assume that the road authority sets the tolls on links 1 *and* 2*, as linear functions of the link flows on the same links, i.e.,* $\theta_1(q_1) = A\,q_1 + B$, $\theta_2(q_2) = A\,q_2 + B$, *where* $\theta_1(\cdot), \theta_1(\cdot) > 0$ *on* $(0, 1)$. *With DUE* $c_1 = c_2 = c_3$ *if all three links are used. The road authority would like the followers to cover the network such that* $q_{l_j}^{(F)} = q_{l_j}^* \quad \forall j \in \{1, 2, 3\}$, *the following linear system has to be solved:*

$$\begin{aligned} \beta_1\, q_1^* + \delta_1 + A\, q_1^* + B &= \beta_2\, q_2^* \\ +\delta_2 + A\, q_2^* + B & \\ \beta_2\, q_2^* + \delta_2 + A\, q_2^* + B &= \beta_3\, q_3^* + \delta_3, \end{aligned}$$

The solution is $A = -1$, $B = 1$. *Thus, if the road authority sets tolls on links* $1, 2$ *as*

$$\theta_1(q_1) = 1 - q_1, \quad \theta_2(q_2) = 1 - q_2,$$

then (q_1^*, q_2^*, q_3^*) *is an optimal response of the travelers and the optimal value of the total travel time for the road authority will be reached.* □

Note that in Example 2.1 the toll is decreasing with the traffic flow rate, such that drivers have incentive to use the link 3, which is untolled, when congestion appears. Other option to obtain the same result would be to toll link 3 with obtaining the same result.

3 Neurosimulation

In this section the application of neurosimulator FAUN in solving the problem (P) will be explained. In the following text the concept of supervision learning will be first introduced, followed by introduction of neurosimulator FAUN.

3.1 Supervised learning

Let function $g : \mathbb{R}^n \to \mathbb{R}^m$ assign to each vector $\mathbf{x}^{(i)} \in \mathbb{R}^n$ a vector $\mathbf{y}^{(i)} \in \mathbb{R}^m$, i.e., $\mathbf{y}_i = g\left(\mathbf{x}^{(i)}\right)$. We will refer to the pair $\left(\mathbf{x}^{(i)}, \mathbf{y}^{(i)}\right)$ as to the i-th *pattern* of the function g. The vector $\mathbf{x}^{(i)}$ will be called the *input* vector (of g) and the vector $\mathbf{y}^{(i)}$ will be called the *output* vector (of g).

Supervised learning constitutes one way of how to find the function g given a set of o patterns (see Knig et al. (2005)). It is of a special interest when other methods like regression methods do not lead to satisfactory results.

An artificial neural network (ANN) can be thought of as a simple mathematical formula with parameters called weights (see Knig et al. (2005) for details). The result of supervised learning applied on function g is an approximation function g^{app} (ANN) with an appropriately chosen vector of weights $\mathbf{w}$. The goal of supervised learning with ANN is to find a function $g^{\text{app}} : \mathbb{R}^n \to \mathbb{R}^m$, which is approximating the function g in the best way.

There are several criteria that can be used to validate whether the function g^{app} is "close enough" to g. In our approach the so-called *validation error* ε_v, introduced below, for each pattern $(\mathbf{x}^i, \mathbf{y}^i)$, $i = 1, 2, \ldots, o$ has to be minimal.

The set of o patterns is divided into a set of t training patterns and a set of $o-t$ validation patterns. The weights of $\mathbf{w}$ are optimized only for the t training patterns, while the validation patterns are used to prevent overtraining. Roughly said: When the training error ε_t becomes small, but the validation error ε_v grows, the ANN learns the patterns "by heart" and looses its interpolation and extrapolation abilities. For a given vector of weights $\mathbf{w}$ the training and the validation errors are calculated with the error functions

$$\begin{aligned} \varepsilon_t(\mathbf{w}) &\stackrel{\text{def}}{=} \frac{1}{2}\sum_{i=1}^{t}\sum_{k=1}^{m}(g_k^{\text{app}}(\mathbf{x}^{(i)};\mathbf{w}) - y_k^{(i)})^2, \\ \varepsilon_v(\mathbf{w}) &\stackrel{\text{def}}{=} \frac{1}{2}\sum_{i=t+1}^{o}\sum_{k=1}^{m}(g_k^{\text{app}}(\mathbf{x}^{(i)};\mathbf{w}) - y_k^{(i)})^2, \end{aligned} \tag{13}$$

where g_k^{app} and $y_k^{(i)}$, $k = 1, 2, \ldots, m$, refer to the k-th entry of g^{app} and $\mathbf{y}^{(i)}$, respectively. An important property of g_{app} is that it has derivatives of all finite orders in the components of $\mathbf{x}^{(i)}$.

An ANN is trained iteratively, i.e., ε_t is decreased by adaption of $\mathbf{w}$, until ε_v increases for two consecutive iterations (prevention of overtraining). The training stops before a local minimum of ε_t is reached. Weight upgrades $\mathbf{w}^{iter+1} - \mathbf{w}^{iter}$ can be calculated with any minimization algorithm, e.g., a first derivative method such as the steepest descent, or a second derivative method such as Newton's method. For first derivative methods the iterative sequence

$$\mathbf{w}^{iter+1} = \mathbf{w}^{iter} + \eta\left(\varepsilon_t\left(\mathbf{w}^{iter}\right), \text{grad}_{\mathbf{w}}\varepsilon_t\left(\mathbf{w}^{iter}\right)\right)\Delta\mathbf{w}\left(\varepsilon_t(\mathbf{w}^{iter}), \text{grad}_{\mathbf{w}}\varepsilon_t\left(\mathbf{w}^{iter}\right)\right), \tag{14}$$

with the search direction $\Delta\mathbf{w}$ and step length η, applies. Numerical methods for constrained nonlinear least-squares problems (see Nowak & Weimann (1998)) are sequential quadratic programming (SQP) methods and generalized Gauss-Newton (GGN) methods, which can exploit the special structure of the Hessian matrix of ε_t (see Deuflhard (2004), Fletcher (2000), Gill et al. (2004)). SQP and GGN methods can automatically overcome most of the training problems of ANN such as flat spots or steep canyons of the error function ε_t. Advantages of these methods are:

- A much better search direction $\Delta\mathbf{w}$ is calculated in comparison to common training methods, e.g., $\Delta\mathbf{w} := \text{grad}_{\mathbf{w}}\varepsilon_t$ for the gradient method (back propagation).
- The step length η is optimized permanently in contrast to common training methods with fixed step length. The number of learning steps is reduced significantly (factor 10 to 1000).
- Only ε_t, $\text{grad}_{\mathbf{w}}\varepsilon_t$, and ε_v are required which mainly can be computed by very fast matrix operations. For other ANN topologies, e.g., radial basis functions, an efficient code for $\text{grad}_{\mathbf{w}}\varepsilon_t$ can also be deduced by automatic differentiation.
- Maximum and minimum of each weight can be set easily (box constraints).
- The total curvature of the ANN can be constrained (prevention from ANN oscillations).
- Convexity and monotonicity constraints can be set.

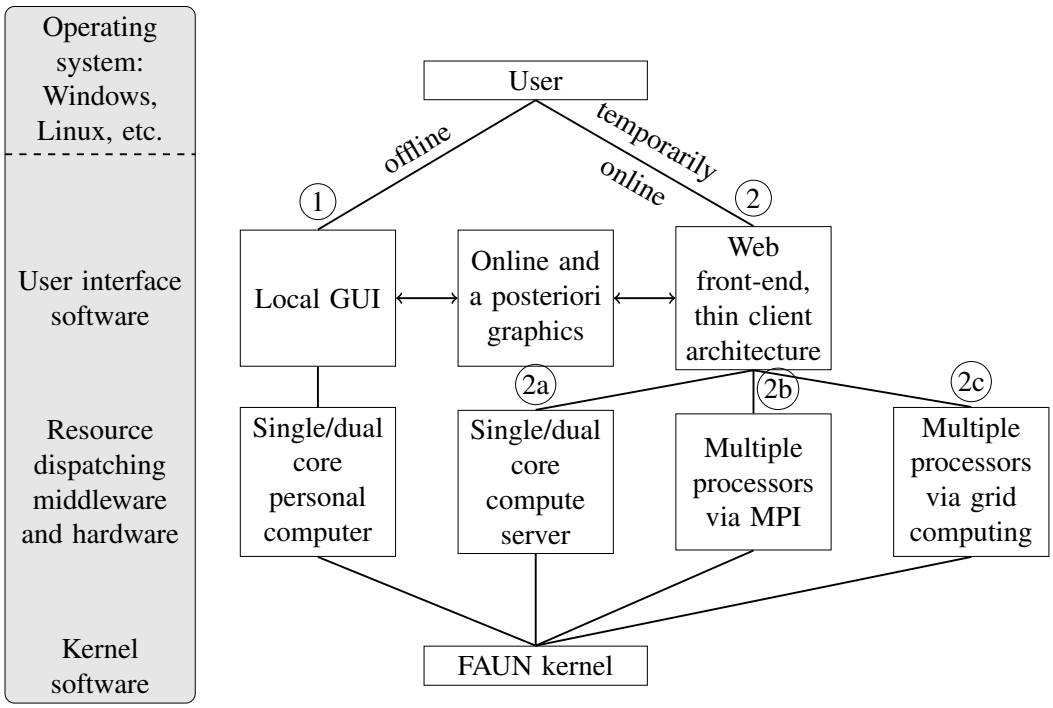


Figure 1: 3-layer architecture of the FAUN software suite. Users choose between local installation (1) or web front-end to access FAUN (2). The middleware distributes tasks user-definable to the FAUN compute kernel on one (2a) ore many processors (2b and 2c). Applications of every layer are independently replaceable and available for Windows and Linux.

3.2 Neurosimulator FAUN 1.1

There are many commercial and public domain (freeware and shareware) neurosimulators, e.g., SNNS, MemBrain, or FANN. The neurosimulator FAUN is portable on Unix, Windows, and other operating systems. The training and learning algorithms of FAUN are based on well-known numerical methods for constrained optimization problems and nonlinear least-squares problems. FAUN has fully automatic prevention from overlearning (adaptable drop out rule). FAUN has an implementation on parallel, vector, and grid computers. FAUN synthesizes functions from high-dimensional input-/output-relations. The architecture of the neurosimular FAUN can be seen in Figure 1. For more information about neural networks and neurosimulator FAUN, see, e.g., Breitner (2004); Breitner et al. (2000); Knig et al. (2005).

4 Solution of the dynamic optimal toll design problem

In this section an algorithm for finding the solution of (P) is presented.

Although the matrix of toll functions Θ defined in (8) can in general consist of any of actual traffic flow rates, in the case studies we will restrict ourselves to linear functions

of link volumes, i.e.,

$$\Theta \stackrel{\text{def}}{=} \begin{pmatrix} \max\left(\gamma_1^1 \frac{x_1^1}{s_1} + \delta_1^1, 0\right) & \dots & \max\left(\gamma_{|\mathcal{A}|}^1 \frac{x_{|\mathcal{A}|}^1}{s_1} + \delta_{|\mathcal{A}|}^1, 0\right) \\ \vdots & \ddots & \vdots \\ \max\left(\gamma_1^{|\mathcal{K}|} \frac{x_1^{|\mathcal{K}|}}{s_1} + \delta_1^{|\mathcal{K}|}, 0\right) & \dots & \max\left(\frac{x_{|\mathcal{A}|}^{|\mathcal{K}|}}{s_{|\mathcal{A}|}} \gamma_{|\mathcal{A}|}^{|\mathcal{K}|} + \delta_{|\mathcal{A}|}^{|\mathcal{K}|}, 0\right) \end{pmatrix}, \quad \gamma_a^k, \delta_a^k \in \mathbb{R}, \tag{15}$$

where the link volume x_a^k [veh] is the number of drivers present on link a at the beginning of the k-th time interval (link volume) and s_a [km] is the length of link a.

We assume that there exist $\gamma_{\min}$, $\delta_{\min}$, $\gamma_{\max}$, $\delta_{\max} \in \mathbb{R}$ such that

$$\gamma_{\min} \leq \gamma_a^k \leq \gamma_{\max}, \quad \delta_{\min} \leq \delta_a^k \leq \delta_{\max}, \quad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}, \tag{16}$$

and that the condition (9) is satisfied.

The algorithm consists of following parts:

- Computing sample points of the total travel time function
  - Outer loop - grid search
  - Inner loop - C-load algorithm for solving the dynamic traffic assignment
- Application of FAUN 1.1 simulator
  - Training of the neural networks and choosing the most suitable candidate
  - Minimizing the function given by the chosen neural network

In the following subsections we will describe individual parts of the solution process.

4.1 Computing sample points of the total travel time function

This algorithm has two built-in optimization procedures: *outer loop* and *inner loop*. Let $n \in \mathbb{N}$ and $m \in \mathbb{N}$ be given. Let us define sets N, M, and $\mathcal{S}^{N,M}$ as follows:

$$N \triangleq \left\{\gamma_{\min}, \gamma_{\min} + \frac{\gamma_{\max} - \gamma_{\min}}{M}, \dots, \gamma_{\max}\right\}, \tag{17}$$

$$M \triangleq \left\{\delta_{\min}, \delta_{\min} + \frac{\delta_{\max} - \delta_{\min}}{N}, \dots, \delta_{\max}\right\}, \tag{18}$$

$$\mathcal{S}^{N,M} \triangleq \left\{\Theta : \forall \gamma_a^k \in N, \quad \forall \delta_a^k \in M \quad \text{satisfying (9) and (16)}\right\}. \tag{19}$$

The set $\mathcal{S}^{N,M}$ will be called the set of *admissible* toll matrices.

In the outer loop of the algorithm the grid search is applied. In each step of the outer algorithm an element of $\mathcal{S}^{N,M}$ is randomly selected and used as an input for the inner loop. By this way a "grid" of sample points of the total travel time is created.

In the inner loop the *dynamic traffic assignment* including *dynamic route choice model*, aiming to determine a stochastic dynamic user-equilibrium based on the actual travel costs, is applied. The C-load algorithm is applied here (Bliemer (2001a)). The dynamic link parameters are recomputed using the *Dynamic network loading* (Chabini (2002)). The DNL model is the heart of the DTA model and is also the most computationally

intensive part. To update route flow rates in each iteration the *method of successive averages* (MSA) is adopted on the route flow level (see Patriksson (1994); Staňková et al. (2008a)). The convergence of the inner loop is verified using so-called *relative dynamic duality gap* $\epsilon^{(i)}$ defined as

$$\epsilon^{(i)} = \frac{\sum_{(r,s)\in\mathcal{RS}} \sum_{p^{(r,s)}\in\mathcal{P}^{(r,s)}} \left(c_p^{(r,s),k,(i)} - \pi^{(r,s),k,(i)} \right) f_p^{(r,s),k,(i)}}{\sum_{(r,s)\in\mathcal{RS}} \pi^{(r,s),k,(i)}} \cdot d^{(r,s),k} \tag{20}$$

Here $\pi^{(r,s),k,(i)}$ is the minimal route travel time for travelers departing from origin r to destination s during the k-th time interval as computed in the i-th iteration. If the relative duality gaps of two consecutive iterations are close enough, i.e., if $|\epsilon^{(i)} - \epsilon^{(i-1)}| < \epsilon_{\max}$ with a given small positive number $\epsilon_{\max}$, the algorithm is terminated.

Pseudocode for computing sample points of the total travel time function

Step 1 (*Initialization*)
Download the network $G(\mathcal{N}, \mathcal{A})$, define $\mathcal{K}$, $\mathcal{RS}$, $\mathcal{P}^{(r,s)}$, $\mathcal{T}$, travel demands, $\epsilon_{\max}$ ($1 \gg \epsilon_{\max} > 0$);
define μ, n, m, N, M, $\mathcal{S}^{N,M}$, $TTT := \infty$;
Define $\epsilon^{(1)}$, $\epsilon^{(0)}$, such that $\left|\epsilon^{(1)} - \epsilon^{(0)}\right| > \epsilon_{\max}$, set network empty.
Step 2 (*Outer loop*: Grid search)
for all N, M, $\Theta \in \mathcal{S}^{\mathcal{N},\mathcal{M}}$ *do*
Step 3 (*Inner loop*: C-load algorithm for Dynamic traffic assignment)
i:=i+1;
while $|\epsilon^{(k+1)} - \epsilon^{(k)}| > \epsilon_{\max}$ do for all $k \in \mathcal{K}$
Step 3a) Compute dynamic link costs from (3) and dynamic route costs from (4);
Step 3b) Determine the route choices of travelers for each $k \in \mathcal{K}$;
Step 3c) Update dynamic route flows using MSA;
Step 3d) Perform DNL to obtain dynamic link flows;
end do;
Compute the total travel time function TTT corresponding to Θ;
Return Θ, TTT;
end do;

4.2 Application of FAUN 1.1 simulator

The grid search produces the value of the total travel time function at discrete positions in the parameter space. A sample is then a combination of toll parameters and corresponding total travel time value.

While it might be possible to try to solve the problem (P) with the grid search only, this approach has two main drawbacks. The grid search is extremely time consuming (if the precise outcome is needed) and the result of the grid search is a set of discrete points, not an analytical description of the function. It would speed up the analysis, if the total travel time could be computed for arbitrary values of the parameter space. This leads to the adopting of the ANN approach.

Pseudocode for applying ANN to the total travel time function

Step 1 (*Initialization*)
Prepare the grid search data for use with FAUN by splitting input and output;
Set appropriate scaling parameters for the data;

Set number of ANN to train successfully N;
Set appropriate worst accepted validation quality;
Prepare FAUN for parallel computation.
Step 2 (*FAUN training*)(Finding appropriate ANN)
do Ntimes in parallel;
Select random W;
while ε_v in (13) does not grow for two consecutive steps *do*
reduce ε_t in (13) by following the gradient descent on W in (14);
end while
if ε_v is acceptable
return and save W;
else if
reinitialize W;
end if
end do
Step 3 (*Postprocessing*)
Export the best ANN using W;
Minimize the ANN;
Return the minimal W and *TTT*.

Since the resulting ANN is smooth and twice differentiable, standard minimization procedures can be applied. In our approach Matlab procedure *fmincon* was used.

The grid search is applied on the neighborhood of the coordinates in which the minimum of the total travel time is reached to check whether the outcome of the algorithm is appropriate. With sufficiently small validation error ε_v this will be always the case.

5 Case studies

In this section case studies with the Chen network consisting of 6 links, 2 origin-destination pairs, and 6 routes will be investigated (depicted in Figure 2). Only link 1 is tollable, the toll is defined by (15).

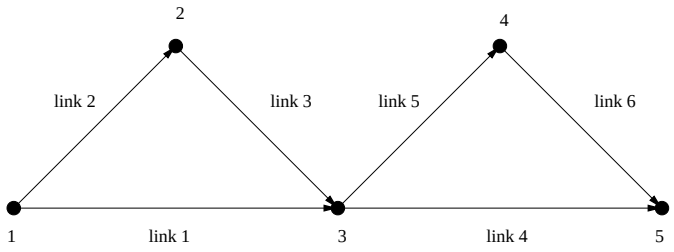
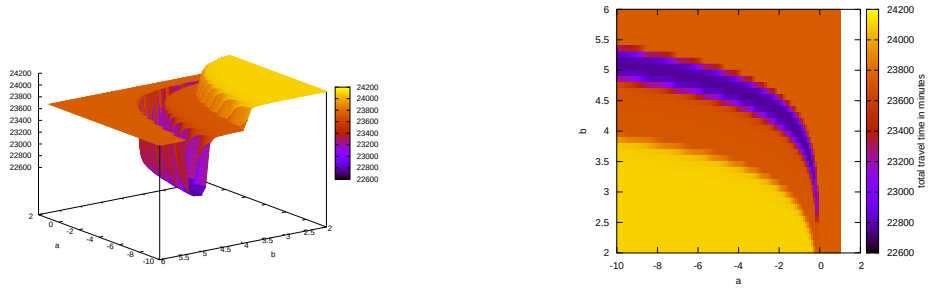


Figure 2: Chen network

Since the problem to be solved is multidimensional and it is hard to see the nonlinearity and nonnconvexity of its objective function, in Figure 3 the total travel time function of the static problem (problem defined for one time interval) defined on the Chen network, with toll given by $\theta_1 \stackrel{\text{def}}{=} a\,\frac{x_1^{(1)}}{s_1} + b = a\,\frac{q_1{}^{(1)}}{s_1} + b$, $a, b \in \mathbb{R}$, $\theta_1 \geq 0$, is depicted. For more information about this problem, see Staňková et al. (2008b).



(a) (b)

Figure 3: Plot and map of objective function of a static problem with toll set to $a \frac{x_1^{(1)}}{s_1} + b$.

5.1 Case study 1

Four time intervals are considered, i.e., $\mathcal{K} = \{1, 2, 3, 4\}$. The link properties and the travel demands are depicted in Table 1. The other parameters are set as: $\mu = 0.2$, $\epsilon = 0.05$, $\alpha = 8$ [€/h], $\gamma_{\min} = -10$, $\delta_{\min} = -5$, $\gamma_{\max} = 10$, $\delta_{\max} = 5$.

Table 1: Link properties and demands in Case study 1.

(a)

a	s_a	$\vartheta_a^{\max}$	ϑ_a^{crit}	$\vartheta_a^{\min}$	J_a^{jam}	C_a
1	7.5	150	90	20	50	1500
2	15	120	70	10	150	3500
3	15	120	70	10	150	3500
4	10	150	90	20	50	1500
5	15	120	70	10	150	3500
6	15	120	70	10	150	3500

(b)

(r,s)	$d^{(r,s),1}$	$d^{(r,s),2}$	$d^{(r,s),3}$	$d^{(r,s),4}$
$(1,5)$	2000	8000	8000	3000
$(3,5)$	1000	1500	2000	1500

The algorithm introduced in the previous section was applied, with 33620 training data, 13297 validation data, and worst accepted validation error equal to 1.1%. Sixteen clusters were used to compute the problem in a parallel way, where both grid search and neurosimulation were parallelized. The neural network function that interpolates the total travel time function the best is smooth, twice differentiable, with more local minima and one global minimum. The minimum $1.4173 \cdot 10^4$ hours was found at $[\gamma_1^1, \delta_1^1, \gamma_1^2, \delta_1^2, \gamma_1^3, \delta_1^3, \gamma_1^4, \delta_1^4]$
$= [-0.50, 0.20, -0.03, 1.19, 0, 0, -0.04, 3.96]$. Note that for the first and forth interval the optimal toll is decreasing with the current traffic volume.

With no toll the total travel time reaches $1.9542 \cdot 10^4$, the optimal time-varying (but traffic-flow independent) tolls are $\theta_1^1 = 2.3$ €,$\theta_2^1 = 6.6$ €, $\theta_2^3 = 9.5$ €, $\theta_1^4 = 7.4$ €, and yield total travel time of $1.7844 \cdot 10^4$ hours.

The computational time of the FAUN simulator was 10.23 hours, the computational time of the grid search was 35.21 hours. This time can be decreased by using more clusters to solve the problem.

5.2 Case study 2

In this case study the number of time intervals will be increased to 8, with travel demands depicted in Table 2.

Table 2: Travel demands - Case study 2

(r,s)	$d^{(r,s),1}$	$d^{(r,s),2}$	$d^{(r,s),3}$	$d^{(r,s),4}$	$d^{(r,s),5}$	$d^{(r,s),6}$	$d^{(r,s),7}$	$d^{(r,s),8}$
$(1,5)$	2000	4000	6000	8000	8000	6000	4000	2000
$(3,5)$	1000	2000	3000	4000	4000	3000	2000	1000

Also, there are no boundaries on parameters of linear toll functions and only 14122 training data and 9301 validation data were used. Worst accepted validation quality was set to 1.1%. The best-trained neural network was minimized using Matlab again. This function is again twice differentiable, with multiple local minima, and one global minimum 29149.00 at $[\gamma_1^1,\ \delta_1^1,\ \gamma_1^2,\ \delta_1^2,\ \gamma_1^3,\ \delta_1^3,\ \gamma_1^4,\ \delta_1^4,\ \gamma_1^5,\ \delta_1^5,\ \gamma_1^6,\ \delta_1^6,\ \gamma_1^7,\ \delta_1^7,\ \gamma_1^8,\ \delta_1^8]$ $= [-0.02,\ 2.62,\ -0.04,\ 3.20,\ 0.4,\ -0.93,\ 0.01,\ -1.32,\ 0.01,\ 0.99,\ 0.05,\ 0.40,\ 0,\ 0,\ 0.02,\ -0.24]$.

Optimal toll decreasing with the current traffic volume appears in the first time interval and in the second time interval. With no toll the total travel time reaches 39659.20 hours. The optimal time-varying (but traffic-flow independent) tolls yield the total travel time of 34822.60 hours.

The computational time of the FAUN simulator was 7.15 hours, the grid search took 26.11 hours. This time can be decreased by using more clusters to solve the problem. From the tests made after the computation it follows that the obtained solution is very accurate in its neighborhood (with an error of 1%), although a lower number of training and validation data was used.

5.3 Discussion

In both case studies the traffic-volume (and hence traffic-flow) dependent toll improved the system performance remarkably. Also, phenomena of the toll decreasing with traffic volume was observed. The natural explanation for this phenomena is that link 1 got very congested and for improvement of the system performance there was a necessity to decrease the traffic on this link.

The grid search is very time consuming, although the network used is very small. The speed of the solution process can be increased by further parallelization of both fazes of the solution process.

Generally, the time-varying but traffic-flow invariant toll can never lead to better outcome than traffic-flow dependent toll. This follows from the fact that the dynamic optimal toll design problem with traffic-flow invariant toll is a special case of (P). See Staňková et al. (2008b) for further explanation.

6 Conclusions & future research

In this paper a solution method for the dynamic optimal toll design problem with the second-best flow-dependent tolling was proposed and presented on small case studies. The problem was solved with use of neurosimulator FAUN 1.1. From a game theory viewpoint the problem of the optimal toll design problem fits within a framework of so-called inverse Stackelberg games.

The proposed solution method is applicable on general traffic networks and in real-time problems of the optimal toll design problem character. However, some issues to speed up the solution process have to be resolved in the case of real-time applications. The mostly computationally intensive part is a grid search to find sufficient number of samples as inputs for the neural network simulation. Parallelization of this process is possible, and it was done also in the case studies presented in this paper. However, much more processors have to be used in the real-time case.

Real-time application of our approach to large-scale problems is being investigated, where the tolls are set as functions of the past traffic flow rates in the network.

In our model the travel demand was assumed fixed. Extension of the existing model to the problem with elastic demand is possible.

Acknowledgments

This research was carried out within the framework of the Next Generation Infrastructures Foundation project and the TRAIL Research School.

The authors are grateful to Institut für Wirtschaftsinformatik, Gottfried Wilhelm Leibniz Universität Hannover, for cooperation and access to the neurosimulator FAUN 1.1.

References

Arnott, R., A. de Palma, R. Lindsey (1990) Economics of a bottleneck, *Journal of Urban Economics*, 27, pp. 11–30.

Başar, T., G. J. Olsder (1999) *Dynamic Noncooperative Game Theory*, SIAM, Philadelphia.

Bagchi, A. (1984) *Stackelberg Differential Games in Economic Models*, Springer-Verlag, Berlin, Germany.

Bard, J. F. (1991) Some properties of the bilevel programming problem, *Journal of Optimization Theory and Applications*, 68, pp. 371–378.

Bliemer, M. C. J. (2001a) *Analytical Dynamic Traffic Assignment with Interacting User-Classes*, Ph.D. thesis, TRAIL Thesis Series, Delft University of Technology, Delft, The Netherlands.

Bliemer, M. C. J. (2001b) *Analytical dynamic traffic assignment with interacting usser-classes*, Ph.D. thesis, The Netherlands TRAIL Research School, Delft, The Netherlands.

Braid, R. (1989) Uniform versus peak-load pricing of a bottleneck with elastic demand, *Journal of Urban Economics*, 26, pp. 320–327.

Breitner, M. H. (2000) Robust optimal onboard reentry guidance of a space shuttle: Dynamic game approach and guidance synthesis via neural networks, *Journal of Optimization Theory and Applications*, 107, pp. 484–505.

Breitner, M. H. (2004) Usage of artificial neural networks for the numerical solution of dynamic games, in: Vincent, T. L., ed., *Proceedings of the Eleventh International Symposium on Dynamic Games and Applications, Tuscon, Arizona*, vol. 1, University of Arizona Press, pp. 62–79.

Breitner, M. H., P. Mehmert, S. Schnitter (2000) Coarse- and fine-grained parallel computation of optimal strategies and feedback controls with multilayered feedforward neural networks, in: Nowak, A., ed., *Proceedings of the Ninth International Symposium on Dynamic Games and Applications, Adelaide, Australia*.

Chabini, I. (2002) Analytical dynamic network loading problem: Formulation, solution algorithms, and computer implementations, *Transportation Research Record: Journal of the Transportation Research Board*, 1771, pp. 191–200.

Defermos, S., F. Sparrow (1971) Optimal resource allocation and toll patterns in user-optimized transport networks, *Journal of transport economics and policy*, 5, pp. 184–200.

Deuflhard, P. (2004) *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms*, Springer, Berlin.

E. T. Verhoef, P. N., P. Rietveld (1996) Second-best congestion pricing: the case of an untolled alternative, *Journal of Urban Economics*, 40(3), pp. 279–302.

Fisk, C. (1980) Some developments in equilibrium traffic assignment, *Transportation Research Part B*, 14, pp. 243–255.

Fletcher, R. (2000) *Practical Methods of Optimization*, John Wiley & Sons, New York.

Gill, P. E., W. Murray, M. H. Wright (2004) *Practical optimization*, Academic Press, London.

Hansen, P., B. Jaumard, G. Savard (1992) New branch and bound rules for linear bilevel programming, *SIAM journal on Scientific and Statistical Computing*, 13, pp. 1194–1217.

Joksimovič, D., M. C. J. Bliemer, P. H. L. Bovy (2004) Optimal toll design problem in dynamic trafic networks-with joint route and departure time choice, *Transportation Research Records*, 1923, pp. 61–72.

Knig, S., F. Kller, M. H. Breitner (2005) *FAUN 1.1 User Manual*, Institut fr Wirtschaftsinformatik, Gottfried Wilhelm Leibniz Universitt Hannover.

Lotito, P., J.-P. Quadrat, E. Mancinelli (2005) Traffic assignment and Gibbs-Maslov semirings, *Contemporary Mathematics*, 377.

Mettenheim, H.-J. v., M. H. Breitner (2005) Neural network forecasting with high performance computers, in: Hofer, E. P., E. Reithmeier, eds., *Proceedings of the Thirteenth International Workshop on Dynamics and Control*, Shaker, Aachen, pp. 33–40.

Mettenheim, H.-J. v., M. H. Breitner (2006) Dynamic games with neurosimulators and grid computing: The game of two cars revisited, in: *Proceedings of the 12th International symposium on dynamic games and applications*, INRIA, France.

Nowak, U., L. Weimann (1998) A family of newton codes for systems of highly nonlinear equations. algorithm, implementation, Tech. Rep. TR 91-10, Konrad Zuse Zentrum, Berlin.

Olsder, G. J. (2005) Phenomena in inverse Stackelberg problems, in: *Regelungstheorie 11*, Mathematisches Forschungsinstitut Oberwolfach, Germany, pp. 603–605.

Patriksson, M. (1994) *The Traffic Assignment Problem: Models and Methods*, VSP, The Netherlands.

Patriksson, M., R. T. Rockefellar (2002) A mathematical model and descent algorithm for bilevel traffic management, *Transportation Science*, 36(3), pp. 271–291.

Smulders, S. (1988) Modelling and filtering of freeway traffic flow, *Report OS-R8706,Centre of Mathematics and Computer Science, The Netherlands*.

Staňková, K., G. Olsder, M. Bliemer (2008a) Dynamic road pricing with traffic-flow dependent tolling, in: *Proceedings of the 87th Transportation Research Board Annual Meeting, CD*, Washington D.C., USA.

Staňková, K., G. Olsder, M. Bliemer (2008b) Flow-dependent tolling in a second-best road pricing: A game theory approach, *submitted to a journal*.

Staňková, K., G. J. Olsder, M. C. J. Bliemer (2006) Bilevel optimal toll design problem solved by the inverse Stackelberg games approach, *Urban transport*, 12, pp. 871–880.

Verhoef, E. T. (2002) Second-best congestion pricing in general networks. heuristic algorithms for finding second-best optimal toll levels and toll points, *Transportation Research Part B*, 36, pp. 707–729.

Vickrey, W. (1969) Congestion theory and transport investment, *The American Economic Review*, 59(2), pp. 251–260.

Wardrop, J. G. (1952) Some theoretical aspects of road traffic research, in: *Proceedings of the Institute of Civil Engineers, Part II*, pp. 325–378.

Yilidirim, M. B., D. W. Hearn (2005) A first best toll pricing framework for variable demand traffic assignment problems, *Transportation Research Part B*, 39, pp. 659–678.