Dynamic Optimal Toll Design Problem with Second-Best-Flow-Dependent Tolling Solved Using Neural Networks

Kateřina Staňková[1], Hans-Jörg von Mettenheim[2],Geert Jan Olsder[3]

ABSTRACT. In this paper the optimal toll design problem as a dynamic game of Stackelberg type is investigated, with the road authority as a leader and drivers of the road network as followers. The road authority tolls 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.

Two types of problems are studied: The "classical" Stackelberg game with the road authority imposing uniform or time-varying tolls, and, as a true extension, the so-called "inverse Stackelberg game" with the road authority setting tolls as functions of traffic flows in the network.

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 is presented on case studies with a small network.

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.

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.

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.

[1]PhD candidate, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, the Netherlands, E-mail: K.Stankova@tudelft.nl.

[2]PhD candidate, Institut für Wirtschaftsinformatik, Gottfried Wilhelm Leibniz Universität Hannover, Germany, E-mail: mettenheim@iwi.uni-hannover.de.

[3]Professor, Faculty of Electrical Engineering, Mathematics and Computer Science, Delft University of Technology, the Netherlands, E-mail: G.J.Olsder@tudelft.nl.

The idea of reducing congestion via appropriate tolls led to the introduction of the so-called *optimal toll design problem* ([1, 2]). This optimal toll design problem is a problem of the Stackelberg type ([3, 4]), 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* ([5, 6]).

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 ([7, 1]). 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 ([8]). 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 ([9]), 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 ([10]).

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 ([1, 11]), with so-called second-best tolling not all links are tollable ([7]). The latter concept is clearly more applicable in practice.

Following extensive case studies of two-route congestion problems in static networks ([12, 13, 14]) 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" ([15, 16]).

This paper introduces an extension of our recent research to dynamic problems with SUE. Although some authors (e.g., [17, 18]) 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 "reasonable" 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., [19, 20, 21].

The contributions of the paper can be listed as follows:

- The dynamic second-best optimal toll design problem with the traffic-flow dependent tolling is formulated. Up to now flow-dependent tolling appeared in static networks, in heuristic studies, or with the first-best pricing only.
- An algorithm solving the problem 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.

INVERSE STACKELBERG GAMES

In this section a basic concept of inverse Stackelberg games with one leader and one follower will be introduced. For more information about (inverse) Stackelberg games we refer to [16, 22].

Let us consider two players, which we call *leader* and *follower*. The leader has decision variable $u_{\mathrm{L}} \in \mathcal{D}_{\mathrm{L}} \subseteq \mathbb{R}$, while the follower has decision variable $u_{\mathrm{F}} \in \mathcal{D}_{\mathrm{F}} \subseteq \mathbb{R}$, where $\mathcal{D}_{\mathrm{L}}$ and $\mathcal{D}_{\mathrm{F}}$ are decision spaces of the leader and the follower, respectively. The leader and the follower have the real-valued cost functions

$$\mathcal{J}_{\mathrm{L}}(u_{\mathrm{L}}, u_{\mathrm{F}}), \quad \mathcal{J}_{\mathrm{F}}(u_{\mathrm{L}}, u_{\mathrm{F}}),$$

respectively. We assume that $\mathcal{J}_{\mathrm{L}}(u_{\mathrm{L}}, u_{\mathrm{F}})$, $\mathcal{J}_{\mathrm{F}}(u_{\mathrm{L}}, u_{\mathrm{F}})$, $\mathcal{D}_{\mathrm{L}}$, and $\mathcal{D}_{\mathrm{F}}$ are known to both players.

Each player chooses her own decision variable in such a way as to minimize her own cost function. Some well-known equilibrium concepts, such as the *Stackelberg equilibrium concept* ([3]), can be used to define a solution. Another equilibrium concept, to be dealt with now, is the *inverse Stackelberg equilibrium*, discussed in, e.g., [15]. The leader does not announce the scalar u_{L}, as above, but a "decision rule" given by the function $\rho_{\mathrm{L}}(\cdot) : \mathcal{D}_{\mathrm{F}} \to \mathcal{D}_{\mathrm{L}}$.

Given the function $\rho_{\mathrm{L}}(\cdot)$, the follower will make her optimal choice u_{F}^* according to

$$u_{\mathrm{F}}^* = \arg\min_{u_{\mathrm{F}} \in \mathcal{D}_{\mathrm{F}}} \mathcal{J}_{\mathrm{F}}(\rho_{\mathrm{L}}(u_{\mathrm{F}}), u_{\mathrm{F}}).$$

The leader, before announcing her $\rho_{\mathrm{L}}(\cdot)$, will realize how the follower will play and she can exploit this knowledge in order to choose the best possible ρ_{L}-function, such that ultimately her own cost function $\mathcal{J}_{\mathrm{L}}$ is minimized. Symbolically, we could write

$$\rho_{\mathrm{L}}^*(\cdot) = \arg\min_{\rho_{\mathrm{L}}(\cdot)} \mathcal{J}_{\mathrm{L}}(\rho_{\mathrm{L}}(u_{\mathrm{F}}^*(\rho_{\mathrm{L}}(\cdot))), u_{\mathrm{F}}^*(\rho_{\mathrm{L}}(\cdot))).$$

In this way one enters the realm of composed functions ([23]), which is known to be a notoriously complex area. From here onward it turns out to be difficult to proceed in an analytic way. However, there is a trick that often works, as shown in Example 2 in [24]. This trick helps the leader to reach the team minimum.

In a Stackelberg setting, the leader acts first and subsequently the follower acts. In an inverse Stackelberg setting, however, while the leader announces her "decision rule", the follower acts first and the leader acts second. A Stackelberg game is a special case of an inverse Stackelberg game with $\rho_{\mathrm{L}}(\cdot)$ chosen as a constant value.

THE OPTIMAL TOLL DESIGN PROBLEM

In this section the dynamic optimal toll design problem is introduced. Under certain conditions, which will be explain in this section, this problem belongs to the class of ISG.

Preliminaries

Let $G(\mathcal{N}, \mathcal{A})$ be a road network defined by a finite nonempty set of nodes $\mathcal{N}$ and a 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 $d^{(r,s),k} \in \mathbb{R}_+$ $[veh/h]$ on the rate 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

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

where the link speed ϑ_a^k [km/h] is defined as *Smulders speed-density function* (see [25]):

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

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

[1]passenger car units

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

where α is the traveler's value of time [$\blacksquare$/h] and θ_a^k [$\blacksquare$] 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, τ_a^k, θ_p^k, θ_a^k, c_p^k, c_a^k are dynamic route travel time for travelers entering route $p \in \mathcal{P}$ during k-th time interval, dynamic link travel time entering link a during k-th time interval, dynamic route toll for travelers entering route $p \in \mathcal{P}$ during k-th time interval, dynamic link toll for travelers entering link a during k-th time interval, dynamic route cost for the travelers entering route p during k-th time interval, and dynamic link cost for travelers entering link a 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 a dynamic link flow rate of travelers entering link a during the k-th interval and $f_p^{k'}$ [veh/h] is a dynamic route flow rate of travelers entering route p during the k'-th time interval. For all $(r,s) \in \mathcal{RS}$, $p \in \mathcal{P}^{(r,s)}$, and $k \in \mathcal{K}$, the *feasibility condition* on route flows has to be satisfied, i.e., vector of route flow rates $\mathbf{f}^{(r,s),k} = \left(f_1^{(r,s),k}, \dots, f_{|\mathcal{P}^{(r,s)}|}^{(r,s),k}\right)'$ belongs to the set $Q^{(r,s),k}$ defined as

$$Q^{(r,s),k} \overset{\text{def}}{=} \left\{ \left(x_1, \dots, x_{|\mathcal{P}^{(r,s)}|}\right)' : \sum_{i \in \left\{1, \dots |\mathcal{P}^{(r,s)}|\right\}} x_i = d^{(r,s),k}, \quad x_i \geq 0, \quad \forall i \in \{1, \dots, |\mathcal{P}^{(r,s)}|\} \right\}. \tag{6}$$

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

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

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 [26] 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. This fits within the framework of *dynamic stochastic user equilibrium model* (see [27, 10]).

Problem formulation

In the inverse Stackelberg setting we, as the road authority, set 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. We assume that these tolls are twice continuously differentiable functions of the link flows. For each time interval the road authority sets the vector of the link tolls $\Theta^k = \left(\theta_1^k(\cdot), \ldots, \theta_{|\mathcal{A}|}^k(\cdot)\right)$, $\theta_a^k(\cdot) : Q^{\mathcal{A},k} \to \mathbb{R}_0^+$, in such a way so as to minimize the total travel time of the system. Trivially, the minimization of the total travel time

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

is equivalent to the minimization of 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

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

subject to DNL, (1) - (5), LB-SUE, (9), while the link and route flows have to be feasible.
Remarks:
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 bilevel programming problems. In [28] it was shown that even the linear bilevel programming problems (with linear objective functions) are NP-hard. Furthermore, in [29] 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. This observation justifies the use of heurictic methods for solving (P), like the neural network approach proposed in this paper. Please note that the problem of finding the uniform toll or time-varying toll is clearly a special case of (P).

NEUROSIMULATION

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

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., $\mathbf{y}_i = g(\mathbf{x}_i)$. We will refer to the pair $(\mathbf{x}_i, \mathbf{y}_i)$ 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 [30]). 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 [30] for details). The result of supervised learning is an approximation function g_{app} with an appropriately chosen vector of weights $\mathbf{w}$. The goal of supervised learning with ANN is therefore to find a function $g_{\text{app}} : \mathbb{R}^n \to \mathbb{R}^m$, which is approximating the function g in the best way. Moreover, it is required that g_{app} has derivatives of all finite orders in the components of $\mathbf{x}$.

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* for every 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. For a given vector of weights $\mathbf{w}$ the training and the validation errors are calculated with the error functions

$$\begin{aligned} \varepsilon_t(\mathbf{w}) &\overset{\text{def}}{=} \frac{1}{2} \sum_{i=1}^{t} \sum_{k=1}^{m} (g_{\text{app},k}(\mathbf{x}_i; \mathbf{w}) - y_{i,k})^2, \\ \varepsilon_v(\mathbf{w}) &\overset{\text{def}}{=} \frac{1}{2} \sum_{i=t+1}^{o} \sum_{k=1}^{m} (g_{\text{app},k}(\mathbf{x}_i; \mathbf{w}) - y_{i,k})^2, \end{aligned} \tag{10}$$

where $g_{\text{app},k}$ and $y_{i,k}$, $k = 1, 2, \ldots, m$, refer to the k-th entry of g_{app} and $\mathbf{y}_i$, respectively. 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 with respect to $\mathbf{w}$, but the validation error ε_v grows, the ANN learns the patterns "by heart" and looses its interpolation and extrapolation abilities.

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). Note that 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 the Newton's method. For first derivative methods we have the iterative sequence

$$\mathbf{w}^{iter+1} = \mathbf{w}^{iter} + \eta\left(\varepsilon_t\left(\mathbf{w}^{iter}\right), \mathrm{grad}_{\mathbf{w}}\varepsilon_t\left(\mathbf{w}^{iter}\right)\right)\Delta\mathbf{w}\left(\varepsilon_t(\mathbf{w}^{iter}), \mathrm{grad}_{\mathbf{w}}\varepsilon_t\left(\mathbf{w}^{iter}\right)\right), \quad (11)$$

with the search direction $\Delta\mathbf{w}$ and with step length η. Numerical methods for constrained nonlinear least-squares problems (see [31]) 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 [32], [33], [34]). It turns out that in practice 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., $\Delta\mathbf{w} := \mathrm{grad}_{\mathbf{w}}\varepsilon_t$ 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.
- Only ε_t, $\mathrm{grad}_{\mathbf{w}}\varepsilon_t$, 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 $\mathrm{grad}_{\mathbf{w}}\varepsilon_t$ 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.

Neurosimulator FAUN

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 is shown in Figure 1.

More detailed information about the neurosimulator Faun can be found, e.g., in [30] and [35].

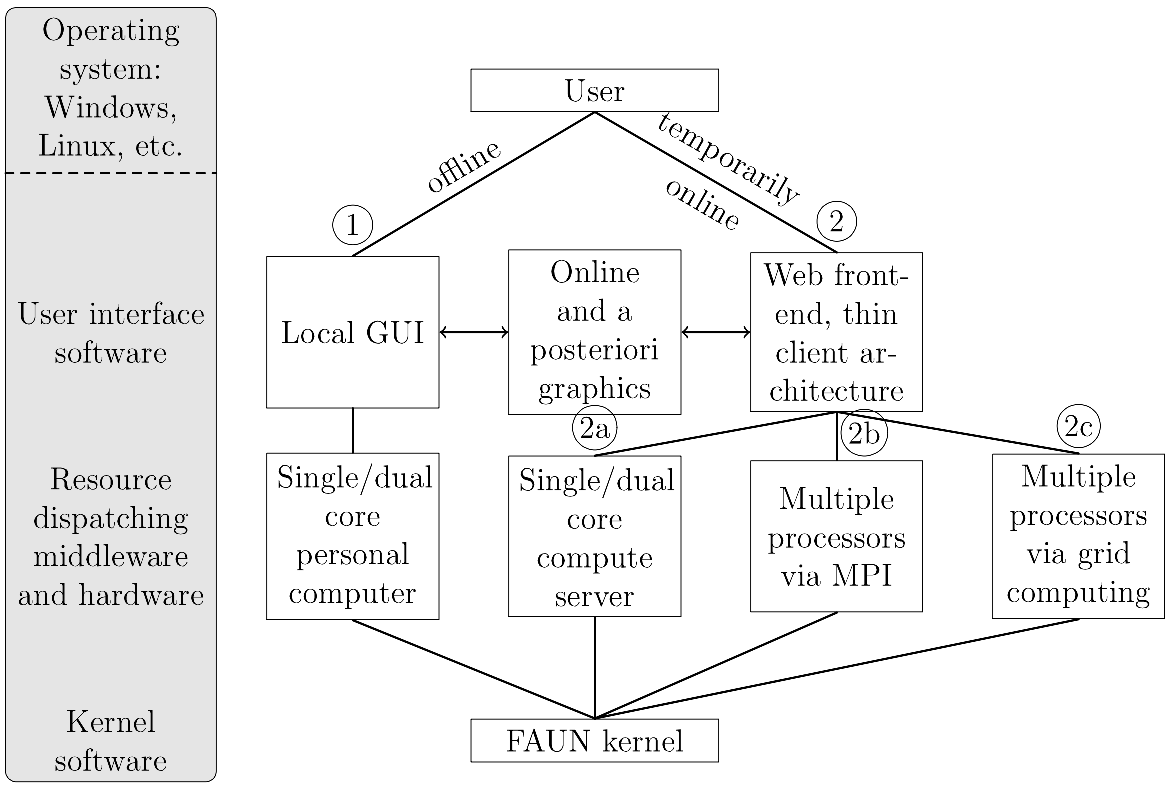

Operating system: Windows, Linux, etc.
User interface software
Resource dispatching middleware and hardware
Kernel software
User
offline
temporarily online
1
2
Local GUI
Online and a posteriori graphics
Web front-end, thin client architecture
2a
2b
2c
Single/dual core personal computer
Single/dual core compute server
Multiple processors via MPI
Multiple processors via grid computing
FAUN kernel

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 microprocessors (2b and 2c). Applications of every layer are independently replaceable and available for Windows and Linux.

SOLUTION OF THE DYNAMIC OPTIMAL TOLL DESIGN PROBLEM

In this section a heuristic algorithm for finding the solution of (P) is presented. Since Stackelberg games can be viewed as a subset of inverse Stackelberg games, we can compare both approaches using the same algorithm that we use for solving the game in the ISG setting.

Although the matrix of toll functions Θ defined in (8) can in general consist of any twice continuously differentiable functions of actual traffic flows, in the case studies we will restrict ourselves to linear functions of link volumes, i.e., with Θ defined as follows:

$$\Theta \stackrel{\text{def}}{=} \begin{pmatrix} \max\left(\gamma_1^1 \frac{x_1^1}{s_1} + \delta_1^1, 0\right) & \dots & \max\left(\gamma_{|\mathcal{A}|}^1 \frac{x_{|\mathcal{A}|}^1}{s_{|\mathcal{A}|}} + \delta_{|\mathcal{A}|}^1, 0\right) \\ \vdots & \ddots & \vdots \\ \max\left(\gamma_1^{|\mathcal{K}|} \frac{x_1^{|\mathcal{K}|}}{s_1} + \delta_1^{|\mathcal{K}|}, 0\right) & \dots & \max\left(\frac{x_{|\mathcal{A}|}^{|\mathcal{K}|}}{s_{|\mathcal{A}|}} \gamma_{|\mathcal{A}|}^{|\mathcal{K}|} + \delta_{|\mathcal{A}|}^{|\mathcal{K}|}, 0\right) \end{pmatrix}, \quad \gamma_a^k, \delta_a^k \in \mathbb{R}, \tag{12}$$

where x_a^k is the number of travelers present on link a at the begining of the k-th time interval (link volume) and s_a is the length of link a.

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

$$\gamma_{\min} \le \gamma_a^k \le \gamma_{\max}, \quad \delta_{\min} \le \delta_a^k \le \delta_{\max}, \quad \forall k \in \mathcal{K}, \forall a \in \mathcal{A}, \tag{13}$$

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 - 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.

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:

$$N \triangleq \left\{\gamma_{\min}, \gamma_{\min} + \frac{\gamma_{\max} - \gamma_{\min}}{n}, \dots, \gamma_{\max}\right\}, \tag{14}$$

$$M \triangleq \left\{\delta_{\min}, \delta_{\min} + \frac{\delta_{\max} - \delta_{\min}}{m}, \dots, \delta_{\max}\right\}, \tag{15}$$

$$\mathcal{S}^{N,M} \triangleq \left\{\Theta : \forall\gamma_a^k \in N, \quad \forall\delta_a^k \in M \quad \text{satisfying (9) and (13)}\right\}. \tag{16}$$

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 dynamic link parameters are recomputed using the *Dynamic network loading* ([26]). The DNL model is the heart of the DTA model and is also the most computationally intensive part. To compute new route flow rates in each iteration the *method of successive averages* (MSA) is adopted on the route flow level (see [5, 24]). The convergence of the inner loop is verified using so-called *relative dynamic duality gap* $\epsilon^{(i)}$ defined as

$$\epsilon^{iter} = \frac{\sum_{(r,s)\in\mathcal{RS}} \sum_{p^{(r,s)}\in\mathcal{P}^{(r,s)}} \left(c_p^{(r,s),k,iter} - \pi^{(r,s),k,iter}\right) f_p^{(r,s),k,iter}}{\sum_{(r,s)\in\mathcal{RS}} \pi^{(r,s),k,(i)}} \cdot d^{(r,s),k} \tag{17}$$

Here $\pi^{(r,s),k,iter}$ is the minimal route travel cost for travelers departing from origin r to destination s during the k-th time interval as computed in the iteration $iter$. If the relative duality gaps of two consecutive iterations are close enough, i.e., if $|\epsilon^{iter} - \epsilon^{iter-1}| < \epsilon_{\max}$ 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 $\mu$, $n$, $m$, $N$, $M$, $\mathcal{S}^{N,M}$, set $TTT_{\min}$ high;,
        Define $\epsilon^1$, $\epsilon^0$, such that $\left|\epsilon^1 - \epsilon^0\right| > \epsilon_{\max}$, set network empty.
Step 2 (*Outer loop*)
        *for* all $N$, $M$, $\Theta \in \mathcal{S}^{\mathcal{N},\mathcal{M}}$ *do*
            iter:=0;
Step 3 (*Inner loop*)(Dynamic traffic assignment)
            iter:=iter+1;
            *while* $|\epsilon^{iter} - \epsilon^{iter-1}| > \epsilon_{\max}$ 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 $\Theta$;
            *if* $TTT < TTT_{\min}$;
                $TTT_{\min} := TTT$;
                $\Theta^* := \Theta$;
            *end if*;
        *end do*;
        Return $TTT_{\min}$, $\Theta^*$;
```

Remarks 1. *The minimization of the total travel time does not need to be done in this stage of computation, but it is useful for future comparison of results computed by neural networks and grid search, respectively.*

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. The ANN approach serves two purposes: first, the grid search is relatively time consuming. It would be nice to have an instantly evaluable function. Second, for every not calculated position in the parameter space the algorithm has to be recomputed. Again, 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 following algorithm.

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 $n_{\mathrm{ann}}$;
        Set appropriate worst accepted validation quality;
        Prepare FAUN for parallel computation.
Step 2 (FAUN training)(Finding appropriate ANN)
        do $n_{\mathrm{ann}}$-times in parallel;
            Select random $\mathbf{w}$;
            while $\varepsilon_v$ in (10) does not grow for two consecutive steps do
                    reduce $\varepsilon_t$ in (10) by following the gradient descent on $\mathbf{w}$ in (11);
            end while
            if $\varepsilon_v$ is acceptable
                    return and save $\mathbf{w}$;
            else if
                    reinitialize $\mathbf{w}$;
            end if
        end do
Step 3 (Postprocessing)
        Export the best ANN using $\mathbf{w}$;
        Minimize the ANN;
        Return the minimal $\mathbf{w}$ and $TTT_{\min}$.
```

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.

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). The Chen network was chosen for its simplicity and also because all important phenomena appearing in more complex networks can be seen in the Chen network, too [36]. Only link 1 is tollable, the toll is defined by (12).

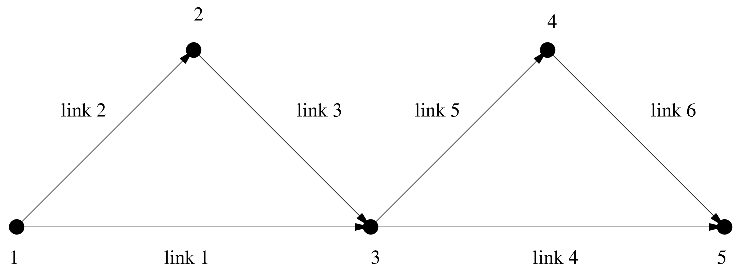
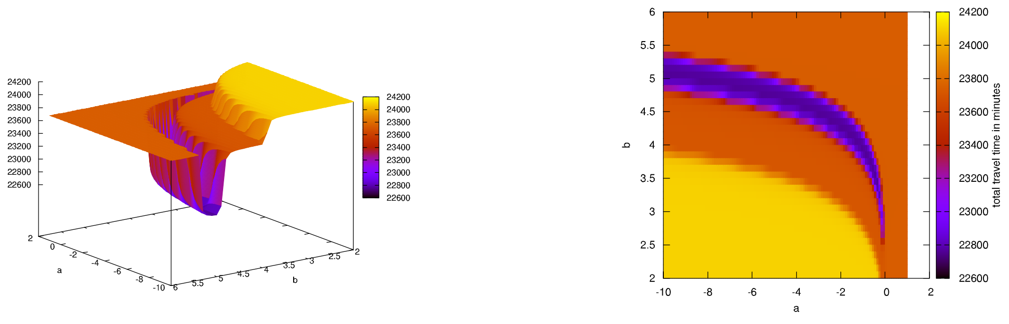


Figure 2: Chen network



(a) (b)

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

To have an impression of what the shape of the objective function, in Figure 3 the objective function of the static problem and with toll defined by (12) as $\theta_1 = a \cdot q_1 + b$, $a, b \in \mathbb{R}$, $\theta_1 \geq 0$ is depicted. Here the points of the total travel time function were computed using grid search.

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, $\gamma_{\min} = -10$, $\delta_{\min} = -5$, $\gamma_{\max} = 10$, $\delta_{\max} = 5$.

The algorithm introduced in the previous section was applied, with 33620 training data, 13297 validation data, and worst accepted validation quality equal to 1.1. Sixteen microprocessors were used to compute the problem in a parallel way. The neural network function that interpolates the total travel time function the best (with validation error 1.5%) has multiple local minima and one global minimum. The minimum $1.4173 \cdot 10^4$ hours at $[\gamma_1^1,\ \delta_1^1,\ \gamma_1^2,\ \delta_1^2,\ \gamma_1^3,\ \delta_1^3,\ \gamma_1^4,\ \delta_1^4] = [-0.50, 0.20, -0.03,\ 1.19, 0, 0, -0.04, 3.96]$ was

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

(a)

(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

(b)

Table 1: Link properties and travel demands in case study 1.

(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

Table 2: Travel demands - case study 2

easily found using the Matlab optimization toolbox. 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 uniform toll is 8.4 ■ and yields a total travel time of $1.7844 \cdot 10^4$ hours.

The computational time of the FAUN simulator was 147676.4 seconds, with the use of 16 microprocessors. However, with a sufficient number of microprocessors the Faun simulator can solve the problem in real time. The most computationally intensive part is the grid search. It takes several hours to compute the sample points. This part of the solution approach can be also run in a parallel way, though.

Case study 2

In this case study the number of time intervals will be increased to 8, with travel demands depicted in Table 2. Also, there are no boundaries on parameters of linear toll functions and only 14122 training samples and 9301 validation samples were used. Worst accepted validation quality was set to 1.1. The best-trained neural network (with validation error 2.5 %) was minimized using Matlab. This function has multiple local minima, and a unique 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 uniform toll is 11.2 ■ and yields a total travel time of 34822.60 hours.

The traffic-flow dependent tolling brought improvement of the total travel time function. The computational time of the FAUN simulator was 1131310 seconds, with the use of 16 microprocessors.

Discussion

In both case studies the traffic-flow dependent toll brought better results than the traffic-flow invariant toll. The algorithm can be applied to bigger networks in real time, provided that the grid search will be parallelized and that more microprocessors will be used.

CONCLUSIONS & FUTURE RESEARCH

In this paper application of Stackelberg and inverse Stackelberg games to the dynamic optimal toll design problem was investigated. The problem was solved with the use of Neurosimulator 1.1. The obtained results suggest that application of traffic-flow-dependent tolling can improve the performance of a traffic system remarkably.

Although with the use of neural networks the computations are rather fast, it is still necessary to speed them up when applying in real-time problems. The computations can be parallelized to achieve faster results, even the real-time computations are possible if many microprocessors would be used. Possible application of our approach to large-scale problems is being investigated.

In our model the travel demand was assumed fixed. Our future research focuses on extension of our existing model to the cases with elastic demand, as well.

References

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

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

[3] Başar, T. & Olsder, G.J., *Dynamic Noncooperative Game Theory.* SIAM: Philadelphia, 1999.

[4] Bagchi, A., *Stackelberg Differential Games in Economic Models.* Springer-Verlag: Berlin, Germany, 1984.

[5] Patriksson, M., *The Traffic Assignment Problem: Models and Methods.* VSP: The Netherlands, 1994.

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

[7] Verhoef, E.T., 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, 2002.

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

[9] Wardrop, J.G., Some theoretical aspects of road traffic research. *Proceedings of the Institute of Civil Engineers, Part II*, pp. 325–378, 1952.

[10] Lotito, P., Quadrat, J.P. & Mancinelli, E., Traffic assignment and Gibbs-Maslov semirings. *Contemporary Mathematics*, **377**, 2005.

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

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

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

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

[15] Olsder, G.J., Phenomena in inverse Stackelberg problems. *Regelungstheorie 11*, Mathematisches Forschungsinstitut Oberwolfach, Germany, pp. 603–605, 2005.

[16] Staňková, K., Olsder, G.J. & Bliemer, M.C.J., Bilevel optimal toll design problem solved by the inverse Stackelberg games approach. *Proceedings of the 12th International conference on Urban transport and the environment in 21st century,*, WIT Press, UK, pp. 871–880, 2006.

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

[18] Arnott, R., de Palma, A. & Lindsey, R., Economics of a bottleneck. *Journal of Urban Economics*, **27**, pp. 11–30, 1990.

[19] Breitner, M.H., 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, 2000.

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

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

[22] Staňková, K. & Olsder, G.J., Inverse Stackelberg games versus adverse-selection principal-agent model theory. *Proceedings of the 12th I-nternational symposium on dynamic games and applications*, INRIA, France, 2006.

[23] Kuczma, M., *Functional Equations in a Single Variable*. Polish Scientific Publishers, 1968.

[24] Staňková, K., Bliemer, M.C.J. & Olsder, G.J., Dynamic road pricing with traffic-flow dependent tolling. *Proceedings of the 87th Transportation Research Board Annual Meeting, CD*, Washington D.C., USA, 2008.

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

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

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

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

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

[30] Knig, S., Kller, F. & Breitner, M.H., *FAUN 1.1 User Manual.* Institut fr Wirtschaftsinformatik, Gottfried Wilhelm Leibniz Universitt Hannover, 2005.

[31] Nowak, U. & Weimann, L., A family of newton codes for systems of highly nonlinear equations. algorithm, implementation. Technical Report TR 91-10, Konrad Zuse Zentrum, Berlin, 1998.

[32] Deuflhard, P., *Newton Methods for Nonlinear Problems. Affine Invariance and Adaptive Algorithms.* Springer, Berlin, 2004.

[33] Fletcher, R., *Practical Methods of Optimization.* John Wiley & Sons, New York, 2000.

[34] Gill, P.E., Murray, W. & Wright, M.H., *Practical optimization.* Academic Press, London, 2004.

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

[36] Chen, K.H., *Dynamic Travel Choice Models: A Variational Inequality Approach.* Springer: Oxford, UK, 1999.