J Glob Optim (2010) 48:533–548
DOI 10.1007/s10898-010-9527-7

A heuristic method for the minimum toll booth problem

Lihui Bai · Donald W. Hearn · Siriphong Lawphongpanich

Received: 4 June 2009 / Accepted: 18 January 2010 / Published online: 5 February 2010

Abstract This paper addresses the toll pricing problem in which the objective is to minimize the number of required toll facilities in a traffic network. The problem is shown to be NP-hard. To obtain a solution in a reasonable time, an effective metaheuristic algorithm is developed. The algorithm uses a local search technique in which the neighborhood function employs the dynamic slope scaling procedure to deal with the fixed charge nature of the objective function. Numerical results from 50 randomly generated and three real networks are reported.

Keywords Congestion pricing · Traffic equilibrium · Dynamic slope scaling

1 Introduction

Today, traffic congestion is a pressing issue for society and a major concern for urban planners. The 2007 Urban Mobility Report [21] states that congestion in the U.S. caused 4.2 billion hours of travel delay and 2.9 billion gallons of wasted fuel for a total cost of $78 billion in 2005. To alleviate congestion, toll pricing has been proposed by transportation economists for over 80 years (see e.g., Pigou [20] and Beckmann et al. [5]). However, it is only

This research was partially supported by NSF grants DMI-9978642, DMI-0300316 and CMS grant #2008-06.

L. Bai (✉)
College of Business Administration, Valparaiso University, Valparaiso, IN 46383, USA
e-mail: lihui.bai@valpo.edu

D. W. Hearn · S. Lawphongpanich
Center for Applied Optimization, Industrial and Systems Engineering Department, University of Florida, Gainesville, FL 32611-6595, USA
e-mail: hearn@ise.ufl.edu

S. Lawphongpanich
e-mail: lawphong@ise.ufl.edu

recently that this idea has been implemented in practice. Examples include the "Electronic Road Pricing" (formerly "Area Licensing Scheme") program in Singapore implemented in 1975, a toll ring in Bergen, Norway implemented in 1986, and subsequently two toll rings in Oslo and Trondheim, respectively. More recently, London introduced a £5 (now raised to £8) daily fee on cars entering the city center in February 2003. In the United States, the Federal Highway Administration (DeCorla-Souza [8]) has been funding toll pricing projects in cities such as San Diego, Houston, and Seattle under the Congestion Pricing Pilot Program established by Congress since 1991. More recently, in Mineta [18], the former U.S. Secretary of Transportation has endorsed the use of congestion pricing in his Six-Point Plan to mitigate our nation's traffic congestion.

This paper focuses on determining tolls that cause travelers, motivated only by their individual travel costs, to choose routes that would collectively benefit all travelers and use the transportation system resources more efficiently. Subject to this requirement, our objective is to find tolls that require the fewest collection facilities or toll booths. This problem was first introduced in Hearn and Ramana [14] and referred to as the minimum toll booth problem (or MINTB). When compared to the marginal social cost pricing (MSCP) advocated by transportation economists, the MINTB solution often requires substantially fewer number of toll booths. For example, the well documented Sioux Falls network [17] with 24 nodes and 76 arcs requires to toll on all 76 arcs under MSCP, but only 32 (see Table 6 in Sect. 5) under the MINTB solution. For the Hull network [9] with 501 nodes and 798 arcs, the MINTB solution requires 39 toll booths and this saves 224 compared to MSCP. Considering that operating a manned toll booth costs $180,000 per year [22], MINTB solutions would save approximately eight and forty million dollars per year for Sioux Falls and Hull, respectively. While today's electronic tolling through pre-registered cards, cameras and other systems can be available at a reasonable cost, which may affect the popularity of manned toll booths, safety concerns still exist when drivers have to merge to electronic tolling lanes frequently. Indeed, the latter may pose a major obstacle to the public and the transportation agency's acceptance of the tolling decisions.

Many works in the congestion pricing literature are related to MINTB in that they also consider to optimize toll levels and toll locations. For example, Yang and Zhang [24] develop a genetic algorithm to determine the optimal toll locations for a "second best" toll pricing problem formulated as a bi-level program. Verhoef [23] proposes a heuristic approach to determine optimal toll levels and toll points so as to maximize the social welfare with elastic demand. Furthermore, Zhang and Yang [27] study optimal toll levels and locations for various topological forms of toll cordons under the cordon pricing system. While the MINTB problem considered in this paper explicitly minimizes the number of required toll booths in the objective function, the above works have objectives of either minimizing the total system travel time with fixed demand or maximizing the social welfare with elastic demand. Consequently, the number of toll locations in [24,27] and [23] is not necessarily the minimum possible.

As stated, the MINTB problem was first introduced as a mixed integer linear programming in Hearn and Ramana [14], and they gave a solution for a nine-node example problem with fixed demand. Subsequently, Yildirim and Hearn [25,26] studied variable demand versions of the problem and solved examples using the same nine-node network. In [14,25] and [26] the nine node MINTB examples were solved with the GAMS [11] modeling language, using CPLEX [7] as the solver. However, as shown in Sect. 3, the MINTB problem is NP-hard. Empirically, numerical results in Sect. 5 confirm that general purpose solvers are not adequately efficient to solve MINTB problems for even moderate size networks such as the aforementioned Hull network. Thus, our focus is to develop heuristic method for fast and "good" MINTB solutions.

The purpose of this paper is to present a heuristic approach, that has been found to be the best for solving larger MINTB problems and to establish empirically when this approach can be preferable to one of the leading mixed integer programming solvers, CPLEX [7]. As the problem size grows, MINTB cannot be solved optimally by any known method in reasonable time, but it is envisioned that the heuristic approach will be valuable in providing good feasible solutions in a situation where transportation planners are experimenting with alternative versions of a real network by varying other network parameters.

For the remainder, Sect. 2 restates the mixed integer program formulation for MINTB in [14], and then the problem is shown to be NP-hard in Sect. 3. Section 4 presents a metaheuristic algorithm using the first improvement local search and a specially designed neighborhood function. Section 5 reports the numerical results on both randomly generated and real transportation networks. Finally, Sect. 6 concludes the paper.

2 Problem formulation

To define the feasible region of MINTB, let $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ be a network where $\mathcal{N}$ and $\mathcal{A}$ represent the sets of nodes and arcs, respectively, in the network, and $\mathcal{K}$ be the set of commodities or origin-destination pairs. For each commodity $k \in \mathcal{K}$, b_k is the demand and $x^k \in R_+^{|\mathcal{A}|}$ is the associated vector of arc flows. Given the latter, $v = \sum_k x^k$ is a vector of total arc flows (or an aggregate flow vector) and the set of all feasible aggregate flow vectors can be written as follows:

$$V^x = \left\{ v : v = \sum_k x^k, Ax^k = b_k E_k, x^k \geq 0, \quad \forall\, k \in \mathcal{K} \right\},$$

where A is the node-arc incidence matrix for network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, and $E_k \in R^{|\mathcal{N}|}$ is a vector with exactly two non-zero elements, one has a value of 1 in the component for the origin of OD-pair k, and the other has a value of -1 in the component for the destination.

It is also possible to describe the above set using path flows. Let f_r denote the flow on route r and P_k represent the set of all routes for OD-pair k. Then, the set of all feasible aggregate flow vectors can be stated as follows:

$$V^f = \left\{ v : v_a = \sum_k \sum_{r \in P_k} \delta_{ar} f_r, \sum_{r \in P_k} f_r = b_k, f_r \geq 0, \quad \forall r \right\},$$

where δ_{ar} is 1 when link a is part of route r and 0 otherwise. When both representations of the feasible region are valid for a discussion, we let V represents either V^x or V^f.

Associated with each arc $a \in \mathcal{A}$, there is also a travel time function, $s_a(v)$. Then, in traffic assignment, an aggregate flow vector v^* is user optimal (UOPT) or in user equilibrium if all the utilized paths (paths with positive flows) associated with v^* have the same travel time that is no longer than those for unutilized paths for any OD-pair. Mathematically, v^* solves UOPT if it solves the following variational inequality, denoted as VI$[s(v), V]$ (see e.g., Florian and Hearn [10]):

$$s(v^*)^T (v - v^*) \geq 0, \quad \forall v \in V.$$

In traffic planning, there is a desired aggregate flow vector, which typically is the system optimal (SOPT) flow, denoted herein as $\bar{v} = \text{argmin} s(v)^T v$. In words, $\bar{v}$ is an aggregate flow vector with the least total travel time. In practice, $\bar{v}$ is generally different from v^*. However, it is possible to charge tolls on links in order to induce users switch to routes that lead to the desired system optimal flow.

Mathematically, let β be a vector of link tolls, Hearn and Ramana [14] consider the tolled user equilibrium problem (TUOPT) or VI$[s(v)+\beta, V]$: $(s(v^*(\beta))+\beta)^T(v-v^*(\beta))\geq 0$, $\forall v \in V$. Note that $s(v)+\beta$ is referred to as the "generalized cost" (time plus tolls). In [14], a toll vector β is valid if $v^*(\beta)=\bar{v}$.

Under mild conditions, Hearn and Ramana [14] show that β is a valid toll vector if and only if there exists a vector of node potentials or multipliers ρ^k such that the following link-based conditions are satisfied:

$$\begin{aligned} s(\bar{v})+\beta &\geq A^T\rho^k, \quad \forall k \in \mathcal{K} \\ (s(\bar{v})+\beta)^T\bar{v} &= \sum_k b_k E_k^T\rho^k. \end{aligned}$$

These conditions are essentially the Karush–Kuhn–Tucker (KKT) conditions (see e.g., Bazaraa et al. [4]) for the TUOPT problem.

Generally, there is no restriction on β. However, negative tolls correspond to subsidies, a politically sensitive topic and a tolling scheme difficult to implement successfully and fairly. In this paper, we assume that β is nonnegative, and consider the minimum toll booth problem (MINTB) that finds a valid toll vector requiring the least number of collection facilities or toll booths. Hearn and Ramana [14] formulate MINTB as follows:

$$\begin{aligned} \text{MINTB:} \quad & \min \sum_{a\in\mathcal{A}} y_a \\ & \text{s.t.} \quad s(\bar{v})+\beta \geq A^T\rho^k \quad \forall\, k \in \mathcal{K} \\ & \qquad (s(\bar{v})+\beta)^T\bar{v} = \textstyle\sum_k b_k E_k^T\rho^k \\ & \qquad 0 \leq \beta_a \leq My_a, \quad \forall\, a \in \mathcal{A} \\ & \qquad y_a \in \{0,1\}, \quad \forall\, a \in \mathcal{A}, \end{aligned} \tag{1}$$

where M is a sufficiently large number. The first two constraints ensure that β is a valid toll vector. The binary variable y_a indicates whether there is a toll booth on arc a. When $y_a=1$, β_a is positive. Otherwise (i.e., $y_a=0$), the corresponding arc toll must be zero.

Although the above link-based formulation of MINTB is more convenient computationally, the path-based formulation defined below is equivalent and facilitates the complexity analysis in the next section. To state the formulation, let P_k^+ and P_k^0 be the sets of utilized and unutilized paths in the system optimal flow, respectively, i.e., $P_k^+=\{r : \bar{f}_r>0, r\in P_k\}$ and $P_k^0=\{r : \bar{f}_r=0, r\in P_k\}$. Here, $\bar{f}_r$ is some system optimal flow on route r, i.e., $\bar{v}_a=\sum_k\sum_{r\in P_k}\delta_{ar}\bar{f}_r$ for all a. Then, the path-based MINTB formulation can be written as follows:

$$\begin{aligned} \text{MINTBP:} \quad & \min \sum_{a\in\mathcal{A}} y_a \\ & \text{s.t.} \quad \textstyle\sum_{a\in r}(s_a+\beta_a)=\lambda_k, \quad \forall\, r\in P_k^+,\ k\in\mathcal{K}, \\ & \qquad \textstyle\sum_{a\in r}(s_a+\beta_a)\geq\lambda_k, \quad \forall\, r\in P_k^0,\ k\in\mathcal{K}, \\ & \qquad 0\leq\beta_a\leq My_a, \quad \forall\, a\in\mathcal{A} \\ & \qquad y_a\in\{0,1\}, \quad \forall\, a\in\mathcal{A}. \end{aligned} \tag{2}$$

In (2), λ_k represents the generalized travel cost for k and the first two conditions ensure that β makes every utilized path under $\bar{v}$ have the same generalized cost that is no more than those for unutilized paths.

3 Computational complexity of MINTB

This section demonstrates that MINTB is NP-hard via the minimum cardinality multi-way cut problem (MCMC). In the literature, MCMC is simply the minimum multi-way cut problem (see e.g., Garg et al. [12] and Costa et al. [6]) in which all edge weights equal one. The problem can be stated as follows:

Minimum Cardinality Multi-way Cut Problem (MCMC) Given a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ with $\mathcal{V}$ and $\mathcal{E}$ being the sets of vertices and edges, respectively, and a set of K pairs of terminal nodes $\mathcal{K} = \{(s_k, t_k),\ k = 1, 2, \dots, K\}$. A *minimum cardinality multi-way cut* is a set of edges with the minimum number of elements whose deletion separates each pair (s_k, t_k), i.e., the remaining graph does not contain any path from s_k to t_k for $k = 1, 2, \dots, K$.

Garg et al. [12] show that MCMC is NP-hard and max SNP-hard when $K \geq 2$. Using this result, we show below that the MINTB problem is also NP-hard. To establish our complexity proof, we restate the minimum toll booth problem in a more convenient form based on the MINTBP formulation at the end of Sect. 2.

Path-based Minimum Toll Booth Problem (MINTBP) Given a directed network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$, an associated travel time vector s_a for all arcs $a \in \mathcal{A}$, and a set of designated utilized paths $P_k^+ \subseteq P_k$ for each OD-pair k. A *minimum toll booth problem* is to find the minimum number of links to charge (positive) tolls so that, for each OD-pair k, all paths in P_k^+ have the same generalized cost that is no larger than those associated with any unutilized path, i.e., paths in $P_k^0 = P_k \setminus P_k^+$.

Below, Theorem 1 establishes the NP-hardness result for the single OD-pair case by reducing the MCMC problem with $K = 2$ to the MINTB problem in polynomial time.

Theorem 1 *The minimum toll booth problem with one OD-pair is NP-hard.*

Proof Consider the MCMC problem with a directed graph $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ where $|\mathcal{E}| = m$ and $\mathcal{K} = \{(s_1, t_1), (s_2, t_2)\}$. Construct a network $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ (see Fig. 1) for the MINTB problem with an OD-pair (o, d), where $o, d \notin \mathcal{E}$, as follows. Let $\mathcal{N} = \mathcal{V} \cup \{o, d\}$ and $\mathcal{A} = \mathcal{E} \cup \{(o, d), (o, s_1)^1, \dots, (o, s_1)^m, (o, s_2)^1, \dots, (o, s_2)^m, (t_1, d)^1, \dots, (t_1, d)^m, (t_2, d)^1, \dots, (t_2, d)^m\}$. Particularly, in addition to (o, d) and those in $\mathcal{E}$, $\mathcal{A}$ contains m parallel arcs between

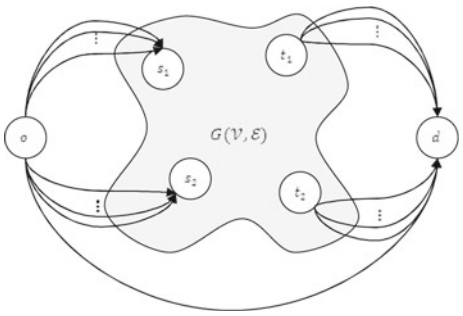

Fig. 1 A MINTB problem with one OD-pair

the pairs (o, s_1), (o, s_2), (t_1, d) and (t_2, d). We refer to these arcs as "connector arcs" and distinguish them by superscript j, $j = 1, 2, \ldots, m$. Set $P^+_{(o,d)} = \{(o, d)\}$, $s_{(o,d)} = 1$ and $s_a = 0$ for all $a \in \mathcal{A}$ and $a \neq (o, d)$. As described, $\mathcal{G} = (\mathcal{N}, \mathcal{A})$ can be constructed in polynomial time.

Let $C^* \subseteq \mathcal{E}$ be a minimum multi-way cut for $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. Then, setting the toll on every arc in C^* to one yields a feasible solution to the MINTB problem because the travel time of every unutilized path is at least one. Thus, $|T^*| \leq |C^*|$, where T^* is the set of toll arcs in an optimal solution to MINTB.

Next we show that $T^* \subseteq \mathcal{E}$. Assume otherwise, which implies that either $T^* \cap \mathcal{E} = \emptyset$ or T^* contains some connector arcs as well as arcs in $\mathcal{E}$. If $T^* \cap \mathcal{E} = \emptyset$, then T^* must consists entirely of connector arcs and it is easy to show that $|T^*| = 2m > |C^*|$, which is a contradiction. When T^* contains some connector arcs as well as arcs in $\mathcal{E}$, let $T^*_{\mathcal{E}} = T^* \cap \mathcal{E}$. Then, $T^*_{\mathcal{E}}$ cannot be a multi-way cut for $\mathcal{G} = (\mathcal{V}, \mathcal{E})$. If so, $T^*_{\mathcal{E}}$ is feasible to MINTB with a smaller number of tollbooths than T^*, which is a contradiction. Since $T^*_{\mathcal{E}}$ is not a multi-way cut, there must exist a toll free path from s_1 to t_1 or from s_2 to t_2. However, the existence of such a path implies that all connector arcs for one of the four pairs, i.e., (o, s_1), (o, s_2), (t_1, d), or (t_2, d), must belong to T^*. Thus, $|T^*| \geq m + |T^*_{\mathcal{E}}| \geq m + 1 > |C^*|$, which is a contradiction.

Given that $T^* \subseteq \mathcal{E}$, it is straightforward to show that T^* must be a multi-way cut for $\mathcal{G} = (\mathcal{V}, \mathcal{E})$, and, because C^* solves MCMC, $|T^*| \geq |C^*|$. Combining this with the first inequality yields that $|T^*| = |C^*|$, i.e., T^* solves MCMC and vice versa (C^* solves MINTB). □

Given that MINTB is NP-hard, it is not surprising that general-purpose mixed integer programming (MIP) solvers would have difficulty with large instances of MINTB and this is confirmed by our results using CPLEX 10.2 [7] reported in Sect. 5. Consideration was given in this study to develop a customized Lagrangian relaxation (see Ahuja et al. [1]) solution procedure. However, it is not expected to perform any better than general-purpose MIP solvers. The result in the "Appendix" shows that the lower bound produced by a Lagrangian relaxation of MINTB is the same as the one obtainable from solving a linear programming relaxation of MINTB, a method used in most MIP solvers. Furthermore, results from a small traffic network in the literature suggest that the bounds so produced are weak. Fortunately, the heuristic approach of the following section has provided an effective method for addressing both randomly generated and realistic MINTB problems. As mentioned previously, the heuristic method is developed using the link-based formulation of MINTB (1) due to its computational efficiency.

4 A dynamic slope scaling procedure based local search for MINTB

This section proposes a local search approach in which a special heuristic, i.e., dynamic slope scaling procedure (DSSP), is used in defining the neighborhood function. Below, we first describe the dynamic slope scaling procedure for MINTB, and then outline the first improvement local search heuristic for MINTB after introduction of the DSSP-based neighborhood function.

4.1 A dynamic slope scaling procedure for MINTB

Rewrite the MINTB problem as follows:

$$\begin{array}{lll} \text{MINTB-FC:} & \min & \sum_{a\in\mathcal{A}} f_a(\beta_a) \\ & \text{s.t.} & s(\bar{v})+\beta \geq A^T\rho^k, \quad \forall\, k \in \mathcal{K} \\ & & (s(\bar{v})+\beta)^T\bar{v} = \sum_k b_k E_k^T \rho^k \\ & & \beta \geq 0, \end{array}$$

where

$$f_a(\beta_a) = \begin{cases} 0, & \text{if } \beta_a = 0 \\ 1, & \text{if } \beta_a > 0. \end{cases}$$

Clearly, $f_a(\beta_a)$ can be viewed as a fixed charge function with the variable cost of 0 and the fixed cost of 1. More importantly, $f_a(\beta_a)$ is concave. Therefore, MINTB-FC is a linearly constrained concave program and must have an optimal solution that is an extreme point. Consequently, there must exist a linear function $h(\beta) = c^T\beta$ such that an optimal solution to the following linear program is also a solution to MINTB-FC:

$$\begin{array}{lll} \text{DSSP-LP:} & \min & h(\beta) \\ & \text{s.t.} & s(\bar{v})+\beta \geq A^T\rho^k, \quad \forall\, k \in \mathcal{K} \\ & & (s(\bar{v})+\beta)^T\bar{v} = \sum_k b_k E_k^T \rho^k \\ & & \beta \geq 0. \end{array}$$

In the literature, Kim and Pardalos [15] solved a fixed charge network flow problem using an algorithm called Dynamic Slope Scaling Procedure (DSSP). To find an appropriate linear function that has the same optimal solution as the fixed charge objective function, Kim and Pardalos [15] start the algorithm with an initial slope c, thus an initial linear objective function $h(\beta) = \beta^T c$. When the resulting linear program is solved, the optimal solution is used to update the slope c, which yields a new (or updated) linear program. This process continues until the change in the two successive optimal solutions is sufficiently small.

Below is the statement of DSSP as it applies to MINTB-FC, which follows the outline described above.

Dynamic Slope Scaling Procedure for MINTB-FC

Step 0 : Let β^1 denote any feasible solution to DSSP-LP and set $l = 2$.

Step 1 : Update c as follows:

$$c_a^l = \begin{cases} \frac{1}{\beta_a^{l-1}} & \text{if } \beta_a^{l-1} > 0 \\ c_a^r & \text{if } \beta_a^{l-1} = 0 \ \text{ and } \ \mathcal{I}(a) \neq \emptyset \\ c_a^1 & \text{if } \beta_a^{l-1} = 0 \ \text{ and } \ \mathcal{I}(a) = \emptyset, \end{cases}$$

where r is the largest, i.e., the most recent, index in set $\mathcal{I}(a) = \{i : \beta_a^i > 0,\ 1 \leq i \leq l-1\}$;

Step 2 : Let β^l solve DSSP-LP with $h(\beta) = \beta^T c^l$. If $\|\beta^l - \beta^{l-1}\| < \epsilon$, stop. Otherwise, set $l = l+1$ and go to Step 1.

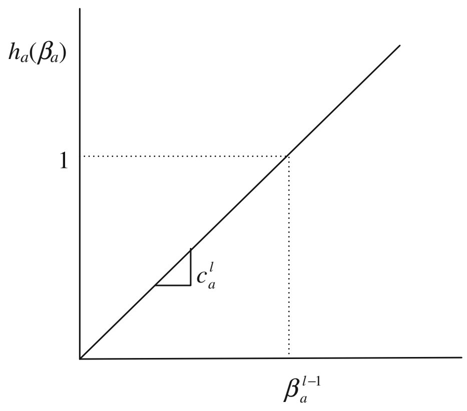

Fig. 2 The slope updating scheme for DSSP

In our implementation, instead of choosing any feasible solution β^1 in Step 0, we set β^1 to be an optimal solution to DSSP-LP with the following slope:

$$c_a^1 = \begin{cases} \frac{1}{s_a'(\bar{v}_a)\bar{v}_a} & \text{if } \bar{v}_a > 0 \\ \frac{1}{2\min\{s_a'(\bar{v}_a)\bar{v}_a : \bar{v}_a > 0\}} & \text{if } \bar{v}_a = 0, \end{cases}$$

where, as before, $\bar{v}$ is the system optimal solution. When $\bar{v}_a > 0$, $s_a'(\bar{v}_a)\bar{v}_a > 0$ is the marginal social cost for link a and the slope c_a^1 is simply the inverse of this cost. On the other hand, when $\bar{v}_a = 0$, the slope for link a is based on the smallest nonzero marginal social cost. Consideration was also given to using the minimum revenue (MINREV) toll (see e.g., Bai et al. [2]) instead of the marginal social cost price (MSCP) toll for computing c^1. However, preliminary results do not indicate much difference between using MINREV and MSCP tolls, thus our implementation used MSCP because it can be easily calculated by a formula.

Figure 2 graphically illustrates the choice of c in Step 1. When $\beta_a^{l-1} > 0$, c_a^l is simply the slope of the line passing through the origin and the point $(\beta_a^{l-1}, 1)$. When $\beta_a^{l-1} = 0$ and $\mathcal{I}(a) \neq \emptyset$, c_a^l equals the most recent slope, c_a^r, a choice suggested by the numerical results in Kim and Pardalos [15]. Observe that the objective function, $h(\beta)$, defined in Step 2 satisfies the following:

$$h_a\left(\beta_a^{l-1}\right) = \beta_a^{l-1} c_a^l = \begin{cases} 1 & \text{if } \beta_a^{l-1} > 0 \\ 0 & \text{if } \beta_a^{l-1} = 0, \end{cases}$$

and $\sum_{a \in \mathcal{A}} h_a(\beta_a^{l-1})$ is the current number of toll booths required. Also, the algorithm terminates in Step 2 when the difference between two successive toll vectors is no larger than ϵ, a small positive constant.

In the literature, properties for the solution of DSSP for fixed-charge network flow problems are discussed. Nahapetyan and Pardalos [19] show that the solution of DSSP can be viewed as that of a related variational inequality. In addition, Lawphongpanich [16] formulates the fixed-charge problem as a mathematical program with complementarity constraints (MPCC) and shows that DSSP is equivalent to solving MPCC using Lagrangian relaxation with subproblem approximation.

The metaheuristic we propose below first implements DSSP to obtain an initial solution, and then performs local search to further improve the solution by DSSP.

4.2 The first improvement local search

In a local search, a neighborhood function determines solutions that are adjacent to the current one. Considering the structure of MINTB and taking advantage of the solution by DSSP, we introduce the *DSSP-based neighbor* for the proposed local search heuristic. Recall that each nonnegative toll vector β is associated with a binary vector y, i.e., $y_a \in \{0, 1\}$, indicating whether link a has a toll booth.

The DSSP-based neighbor is a feasible toll vector generated from the current toll vector by closing the toll booth on one currently tolled arc. Mathematically, for a given solution $\hat{y}$, let $\Omega(\hat{y}) = \{a : \hat{y}_a = 1\}$ be the set of arcs with toll booths. Then, y is a *DSSP-based neighbor* of $\hat{y}$ if it solves the following (restricted) MINTB problem with $y_i = 0$ for some $i \in \Omega(\hat{y})$. Thus, the neighborhood function for $\hat{y}$ is defined as $\mathcal{N}(\hat{y}) = \{y | y = \text{argmin (RMINTB) for some } i \in \Omega(\hat{y})\}$.

$$
\begin{array}{lll}
\text{RMINTB:} & \min & \sum_{a \in \mathcal{A}} y_a \\
& \text{s.t.} & s(\bar{v}) + \beta \geq A^T \rho^k, \quad \forall\, k \in \mathcal{K} \\
& & (s(\bar{v}) + \beta)^T \bar{v} = \sum_k b_k E_k^T \rho^k \\
& & 0 \leq \beta_a \leq M y_a, \quad \forall\, a \in \mathcal{A} \\
& & y_a \in \{0, 1\}, \quad \forall\, a \in \mathcal{A} \\
& & y_i = 0.
\end{array}
$$

For example, if $\hat{y} = (1, 0, 1, 1, 0)$, then $\hat{y}$ has at most 3 DSSP based neighbors obtained by solving RMINTB three times with $y_1 = 0$, $y_3 = 0$, $y_4 = 0$, respectively. It should be noted that RMINTB may not be feasible by imposing constraint $y_i = 0$. In the numerical implementation, we used DSSP to solve RMINTB heuristically.

The DSSP-based neighborhood function only produces the deletion type of neighbors. The reason we do not choose the insertion neighborhood function, i.e., open some currently closed toll booths, is the following. First, the insertion type function would duplicate the work already performed by DSSP. Second, our experience with MINTB suggests that in most cases the number of toll booths is small relative to the total number of arcs. Thus, the size of the insertion neighborhood is usually larger than that of the deletion neighborhood. Therefore, the resulting search would be quite expensive.

Given the neighborhood function, below we outline the local search algorithm for MINTB. The algorithm iteratively chooses the first improved neighbor to replace the current best solution until no improvement can be made.

First Improvement Local Search (FILS) for MINTB

Step 1: Compute an initial toll by DSSP, i.e., $\beta^1 = \beta_{\text{dssp}}$ and set y^1 accordingly. Set $l = 1$.
Step 2: Set $\Omega(y^l) = \{a : y_a^l = 1\} = \{a_1, a_2, \ldots, a_{|\Omega^l|}\}$ and $i = 1$.
Step 3: Let $\bar{y} = \text{argmin(RMINTB)}$. If RMINTB is feasible and $\sum_a \bar{y}_a < \sum_a y_a^l$, then let $y^{l+1} = \bar{y}$ and $l = l + 1$ and go to Step 2. Otherwise, go to Step 4.
Step 4: If $i = |\Omega^l|$ then stop and accept $y^* = y^l$ as a solution. Otherwise, $i = i + 1$ and go to Step 3.

5 Numerical results

In this section, we discuss results from our numerical experiments using the proposed heuristic FILS. The CPU times reported here are from a Dell PowerEdge 2600 with dual Pentium

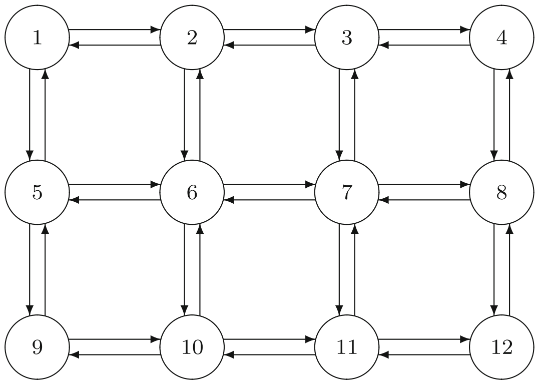

Fig. 3 A grid network

Table 1 Random networks attributes and MINTB problems sizes

Set index	# of nodes in each row	# of nodes in each column	Total # of nodes	Total # of arcs	Total # of commodities	Total # of var.	Total # of constr.
1	10	5	50	170	5	590	1,021
2	10	10	100	360	10	1,720	3,961
3	10	15	150	550	15	3,350	8,801
4	10	20	200	740	20	5,480	15,541
5	10	25	250	930	25	8,110	24,181

3.2 GHz Xeon processors and 6 GB of RAM. The algorithm is implemented in GAMS [11], a modeling language, and we used CPLEX 10.2 [7] to solve DSSP-LP. For comparison purposes, we also solved all problems using CPLEX 10.2's mixed integer solver. Finally, we terminated CPLEX and FILS after a maximum CPU limit of 100,000 s, approximately 1.2 days.

We used two sets of networks in our experiments. One set consists of 50 randomly generated grid networks. Figure 3 displays a grid network of size 3×4. These grid networks are meant to model the downtown or business center areas in cities, which traffic planners may extract from the entire network for a more detailed investigation. In general, the grid networks representing these areas tend to be dense. The travel time functions for the random networks here are of the form $s_a(v_a) = F_a(1 + 1.5 \times (v_a/C_a)^4)$, where both F_a and C_a are random numbers in the interval [1,10]. Similarly, the travel demand for each commodity or OD-pair is a randomly chosen number in interval [5,15]. For our experiments, we generated five sets of random grid networks with attributes listed in Table 1. Also included in Table 1 is the MINTB problem size for each set of networks, i.e., total number of decision variables and total number of constraints. Finally, note that the numbers of binary variables and fixed charge constraints both equal the number of arcs in the network. Thus, they are 170, 360, 550, 740, and 930 for sets 1–5, respectively. For the smallest set 1, we tested ten instances to compare FILS and the mixed integer solver of CPLEX. For sets 2–5, because CPLEX took too long to solve the problems optimally we only tested 5 instances for each set.

For the ten random networks in Set 1, Table 2 compares the proposed local search heuristic, i.e., FILS, with CPLEX's mixed integer solver. It contains information on performance measures for both solvers as follows. For the heuristic, the number of toll booths used in the

Table 2 Results for random networks: set 1

Inst. Ind.	DSSP		FILS			DSSP-FILS	CPLEX			# T.B.
	# T.B.	CPU	# T.B.	CPU_{bi}	CPU_{te}	CPU	# T.B.	CPU_{bi}	CPU_{te}	MSCP
1.1	43	0.23	43	0	7.5	7.73	42	42.96	83.7	107
1.2	39	0.19	37	2.27	6.99	7.18	34	2.4	2.4	89
1.3	47	0.26	46	7.39	19.33	19.59	43	663.99	6,206.4	109
1.4	31	0.23	31	0	4.72	4.95	30	15.81	34.0	87
1.5	31	0.18	30	3.21	7.86	8.04	30	16.94	38.6	81
1.6	49	0.18	48	1.76	10.88	11.06	46	205.68	1,290.4	118
1.7	23	0.15	23	0	2.22	2.38	23	5.6	5.6	66
1.8	39	0.30	39	0	6.9	7.2	39	22.16	92.7	97
1.9	45	0.12	43	1.24	7.89	8.01	43	239.88	48,885.2	108
1.10	41	0.15	40	1.52	6.07	6.22	39	37.58	77.1	102

solution and the computational CPU time are reported for both the initial DSSP and FILS. Particularly, for FILS we report the CPU time used to locate the best incumbent solution as well as the total execution time denoted by CPU_{bi} and CPU_{te}, respectively. In other words, CPU_{bi} excludes the time used in the final iteration of the neighborhood search when no improvement is found. Additionally, the total CPU time combining the initial DSSP and local search procedures is listed under column "DSSP-FILS CPU." Similarly, the "CPU_{bi}" for "CPLEX" gives the CPU time used by CPLEX to locate the best incumbent solution excluding the time used for proving optimality. Furthermore, we also include the number of toll booths and the total execution time (CPU_{te}) for CPLEX solutions. Finally, the last column reports the number of toll booths required by the marginal social cost prices (MSCP).

From Table 2 we find that for all ten instances in the smallest random network set, CPLEX was able to obtain optimal solutions within the CPU limit, which can be used as benchmarks for our heuristic solutions. Note that for instances 1.1, 1.4, 1.7, and 1.8, the local search did not improve the DSSP heuristic solution, thus the best-incumbent CPU time for FILS is reported as zero. Among the ten, the best cases are instances 1.5, 1.7, 1.8, and 1.9 where the heuristic method produced optimal solutions with much less CPU time. For example, when examining the total execution time, DSSP-FILS used 8.04 s to obtain an optimal solution with 30 toll booths for instance 1.5, while CPLEX used 38.6 s, approximately nine times of the CPU time used by the heuristic, for the same optimal solution. Notice that instance 1.9 could be an outlier where CPLEX took enormous time (48,885.2 s) to obtain an optimal solution compared to the proposed heuristic (7.89 s). Consistently, when examining the CPU time used for locating the best incumbent solution excluding the time for proving local (for DSSP-FILS) or global (for CPLEX) optimality, DSSP-FILS also outperforms CPLEX. For the other six instances where our heuristic terminated rather sooner returning only a local optimal solution, the optimality gaps of the heuristic solutions are 2.3, 8.8, 6.9, 3.3, 4.3, and 2.6% for instances 1.1, 1.2, 1.3, 1.4, 1.6, and 1.10, respectively. The average optimality gap among the six is 4.7%. Finally, it is worth noting that the numbers of toll booths required by MSCP are at least twice as many as the ones required by the MINTB solutions from either the heuristic or CPLEX.

Furthermore, Table 3 indicates that random networks in sets 2 and 3 pose difficult MINTB problems in that CPLEX could not solve these problems to optimality within the CPU limit of 100,000 s. Thus, compared to Table 2, Table 3 replaces the total execution time ("CPU_{te}")

Table 3 Results for random networks: sets 2–3

Inst. Ind.	DSSP		FILS			DSSP-FILS	CPLEX			# T.B.
	# T.B.	CPU	# T.B.	CPU$_{bi}$	CPU$_{te}$	CPU	# T.B.	CPU$_{bi}$	Opt. gap (%)	MSCP
2.1	122	14.28	118	22.8	720.48	734.76	**118**	64,829.26	19.6	285
2.2	131	3.96	129	57.45	645.40	649.36	**129**	34,862.74	20.5	303
2.3	105	0.69	102	66.2	153.79	154.48	**101**	77,126.00	14.7	254
2.4	112	5.86	111	26.47	463.81	469.67	**110**	69,374.34	16.6	245
2.5	134	4.22	130	119.41	643.14	647.36	137	61,928.28	25.5	280
3.1	194	14.95	189	7,794.71	12,542.98	12,557.93	190	268.76	42.55	408
3.2	208	44.20	203	1,036.99	11,054.28	11,098.48	**203**	2,751.63	34.50	443
3.3	208	13.36	204	1,291.72	5,860.26	5,873.62	**201**	12,457.71	30.57	443
3.4	186	11.89	183	3,209.13	6,099.28	6,111.17	184	10,124.93	32.18	418
3.5	215	42.97	210	1,800.11	7,414.41	7,457.38	228	65,187.82	38.58	455

Bold values for # T.B. under column “CPLEX” indicate CPLEX obtained the same or better objective values than FILS

with the optimality gap (“Opt. gap”) information for CPLEX solutions. From Table 3, the optimality gap for CPLEX’s incumbent solutions ranges from 15 to 26% for networks in set 2, and from 31 to 43% for those in set 3. Comparing the performance of the two solvers, one finds that CPLEX provided comparable or better solutions (meaning the solution uses the same or less number of toll booths) than the heuristic method for four out of five networks in set 2. However, this advantage diminishes as the problem sizes increases in set 3, where CPLEX produced comparable or better solutions only for two out of five networks. For example, for instance 3.2 the heuristic used approximately 1,080 $(1036.99 + 44.2)$ s to locate a solution with 203 toll booths and 11,055 s to complete the search, while CPLEX found a solution with the same number of toll booths after 2,752 s and made no improvement before reaching the limit of 100,000 s. As another example, CPLEX took 65,188 s to find a solution using 228 toll booths for instance 3.5; the heuristic, on the other hand, located a solution using 18 fewer (210) toll booths in roughly 1,843 $(1800.11 + 42.97)$ s, 2.8% of the time used by CPLEX. Note that when comparing the best-incumbent CPU time, instance 3.1 is an exception where CPLEX was quicker than DSSP-FILS. In this case, CPLEX identified a solution using 190 toll booths in 268.76 s while DSSP-FILS used 7,810 $(7794.71 + 14.95)$ s to obtain a solution with 189 toll booths.

Our numerical results on the two largest random network sets 4 and 5 are reported in Table 4. They confirm that as the problem size increases the heuristic becomes more advantageous. Among the ten instances from both sets, CPLEX was able to find an incumbent solution quicker for two cases: instances 5.1 and 5.5. However, the CPLEX solutions use one and five more toll booths than the ones by DSSP-FILS for instances 5.1 and 5.5, respectively. For all eight other instances from both sets, the heuristic method DSSP-FILS provided better solutions than CPLEX in much less CPU time, considering both the best-incumbent and total-execution CPU times.

The other set of test problems consists of real traffic networks from the literature and their attributes are listed in Table 5. As in Table 1, the numbers of binary variables and fixed charge constraints are not listed because they are the same as the number of arcs in the network. That is, the numbers of binary variables or fixed charge constraints are 76, 798 and 962 for Sioux Falls, Hull and Stockholm, respectively.

Table 4 Results for random networks: sets 4–5

Inst. Ind.	DSSP		FILS			DSSP-FILS	CPLEX			# T.B.
	# T.B.	CPU	# T.B.	CPU$_{bi}$	CPU$_{te}$	CPU	# T.B.	CPU$_{bi}$	Opt. gap (%)	MSCP
4.1	265	42.19	261	15,289.46	27,424.11	27,466.3	272	31,122.65	49.12	591
4.2	290	36.33	284	15,390.16	27,582.53	27,618.86	289	81,869.10	49.13	608
4.3	276	25.39	271	5,780.84	12,983.33	13,008.72	273	25,322.62	42.06	598
4.4	316	38.57	310	3,599.57	60,416.58	60,455.15	313	12,769.30	40.43	651
4.5	293	220.75	288	12,235.51	30,723.15	30,943.9	294	84,330.49	40.70	631
5.1	382	515.73	376	85,917.96	Limit reached	Limit reached	377	44,214.29	48.25	812
5.2	321	55.45	319	1,091.31	17,319.61	17,375.06	323	52,125.46	39.30	709
5.3	360	483.39	354	18,055.75	90,876.8	91,360.19	357	51,489.30	43.24	748
5.4	397	310.83	390	16,063.0	87,154.39	87,465.22	394	53,871.09	49.45	792
5.5	373	129.60	369	41,323.95	Limit reached	Limit reached	374	12,299.45	48.71	792

Table 5 Network attributes and MINTB problem sizes

Network	Links	Nodes	Commodities	# of var.	# of constr.
Sioux Falls [17]	76	24	528	92,928	40,205
Hull [9]	798	501	138	289,386	110,923
Stockholm [14]	962	416	1,623	3,797,820	1,562,289

Table 6 Results for real networks

Real networks	DSSP		FILS			DSSP-FILS	CPLEX			# T.B.
	# T.B.	CPU	# T.B.	CPU$_{bi}$	CPU$_{te}$	CPU	# T.B.	CPU$_{bi}$	Opt. gap (%)	MSCP
Sioux Falls	38	0.5	35	0.86	22.79	23.29	32	41.68[a]	0	76
Hull	42	3.27	41	28.72	162.65	165.92	39	29,344.77	14.1	263
Stockholm	134	92.71	130	6,658.69	20,819.68	20,912.39	111	4,185.38	57.76	808

[a] The total execution CPU time is 156.2 s

Table 6 compares FILS and CPLEX using the same measurements as in previous results tables. For Sioux Falls, the heuristic used 23.29 s to obtain a solution with 35 toll booths, while CPLEX used 156.2 s for an optimal solution with 32 toll booths. When comparing the CPU times for locating these solutions, DSSP-FILS used 1.36 $(0.5 + 0.86)$ s and CPLEX used 41.68 s. Overall, the heuristic method provided a solution with 9.38% optimality gap in approximately seven times less CPU time. Note that the number of toll booths required by the marginal social cost prices (MSCP) is 76, compared to 32 by CPLEX and 35 by the heuristic. For Hull, CPLEX terminated after reaching the CPU limit of 100,000 s with an incumbent solution using 39 toll booths found in approximately 29,345 s. Within the 100,000 CPU seconds, CPLEX reported a 14.1% optimality gap for this incumbent solution. On the other hand, the heuristic found a solution with 41 toll booths in 31.99 (3.27 + 28.72) s and completed the search in 165.92 s. For Stockholm, CPLEX also terminated due to the CPU limit of 100,000 s, and it outperformed DSSP-FILS in both the best-incumbent CPU time and solution quality. Note again that MSCP requires substantially larger number of toll booths than does the MINTB solution (either from the heuristic or CPLEX) for both Hull and Stockholm.

Summarizing the results for both random and real networks, we offer several remarks. First, the heuristic method has its advantage when dealing with dense and larger networks. While it can be argued that CPLEX could provide better or even optimal solution given enough solution time and memory, our computational experience suggests that CPLEX progresses extremely slowly after a certain period, partially due to the rather weak lower bound for MIPs that involve fixed charge constraints such as MINTB. This implies that more investment on time and memory may not be paid off, thus using a heuristic method becomes appealing. Second, in most cases DSSP alone provides fast and feasible solutions with a reasonable optimality gap, thus it can be useful for traffic planners who are altering other network parameters while also wanting to know the minimum number of toll booths quickly. Third, the proposed local search heuristic method can be used as a pre-processing tool to provide an initial feasible solution for an exact solution method. The heuristic will provide a fairly tight upper bound for the exact method, thus expediting the overall solution process. The effectiveness of this proposal could be investigated in a future research. Finally, solving MINTB problems for networks as large as Stockholm remains to be a challenge. CPLEX returned a solution with a 58% optimality gap after reaching the CPU limit of 100,000 s, while our heuristic terminated after 21,000 CPU seconds with only a local solution.

6 Conclusions

In this paper, we establish that the MINTB problem is NP-hard and present a meta-heuristic algorithm based on the dynamic slope scaling procedure using first improvement local search. The method is effective at solving the MINTB problems from the literature and the randomly generated networks. Especially, the method shows advantages for large and dense networks, as numerical results indicate general MIP solvers such as CPLEX 10.2 would take considerably more CPU time to produce solutions comparable to those from the proposed heuristic. Furthermore, the proposed heuristic or the DSSP procedure alone can be useful for traffic planners who wish to experiment with other parameter settings for a real network. Finally, our preliminary test shows that using DSSP to warm start CPLEX has reduced the total solution time noticeably (an average of approximately 36% less CPU time among ten random instances in Set 1 of the grid networks), whereas using the readily available MSCP to initialize CPLEX increases the total solution time rather marginally (an average of roughly 2% more CPU time among ten instances in Set 1 of the grid networks). Therefore, using the proposed heuristic (or variations such as ADCUP [19]) as a pre-processing tool to expedite a customized exact solution method (see e.g. Bai and Rubin [3]) can be an interesting topic for future research.

Appendix

Consider the following MINTB problem in which all tolls are nonnegative:

$$\begin{array}{rll} \text{A: } z = \min & \sum\limits_{a\in\mathcal{A}} y_a & \\ \text{s.t.} & s(\bar{v})+\beta \geq A^T\rho^k, & \forall\, k \in \mathcal{K} \\ & (s(\bar{v})+\beta)^T\bar{v} = \sum_k {b_k}^T\rho^k & \\ & 0 \leq \beta_a \leq My_a, & \forall\, a \in \mathcal{A} \\ & y_a \in \{0, 1\}, & \forall\, a \in \mathcal{A}, \end{array}$$

where M is a sufficient large number. Let z_{LP} denote the optimal objective value of problem A when the last set of constraints is replaced by $0 \leq y_a \leq 1, \forall a \in \mathcal{A}$. In other words, z_{LP} is the optimal objective value for the linear programming (LP) relaxation of problem A.

A Lagrangian dual function for problem A can be written as follows:

$$\begin{array}{rcll} \text{B: } L(\lambda) & = & \min\limits_{y,\beta,\rho} & \sum_a y_a + \sum_a \lambda_a(\beta_a - My_a) \\ & & \text{s.t.} & s(\bar{v}) + \beta \geq A^T\rho^k, \quad \forall\, k \in \mathcal{K} \\ & & & (s(\bar{v}) + \beta)^T\bar{v} = \sum_k {b_k}^T\rho^k \\ & & & \beta_a \geq 0, \quad \forall a \in \mathcal{A} \\ & & & y_a \in \{0, 1\}, \quad \forall a \in \mathcal{A}. \end{array}$$

where λ_a is the dual multiplier associated with the constraint $\beta_a \leq My_a$. Let L^* denote $\max\limits_{\lambda\geq 0} L(\lambda)$. Then, the following result implies that the lower bounds from the Lagrangian dual function and LP relaxation are the same.

Theorem 2 $L^* = z_{\mathrm{LP}}$

Proof By writing the objective function of problem B in the following form,

$$\min_{y,\beta,\rho} \sum_a (1 - M\lambda_a)y_a + \sum_a \lambda_a\beta_a$$

it is clear that the decision variables y and (β, ρ) are independent. From the above, it is also clear that the optimal value for y_a is naturally binary, i.e., $y_a = 1$ when $(1 - M\lambda_a) < 0$ and zero otherwise. So, the optimal solution to problem B remains the same if its last set of constraints is replaced by $0 \leq y_a \leq 1, \forall a \in \mathcal{A}$. Doing so makes $L(\lambda)$ also a Lagrangian dual function for the LP relaxation of problem A. Because linear programs have no duality gap, $L^* = z_{\mathrm{LP}}$. □

To test the quality of the LP or, equivalently, Lagrangian relaxation lower bound for MINTB, consider the *nine-node* network from Hearn and Ramana [14]. The network consists of nine nodes and 18 links. The optimal MINTB solution requires five toll booths with the maximum toll of 11.20. Our experiment shows that the quality of the lower bound varies drastically with different choices of M. When M equals 50, L^* equals 0.592, approximately 11.4% of the optimal number of toll booths. When M equals 12, a value slightly larger than the maximum toll (11.20), L^* increases to 2.467, a rather poor lower bound considering the optimal number of toll booths is five. Since the nine-node problem has been used in the literature for testing algorithms (see e.g., Hearn et al. [13]) in transportation science, it is not expected that LP or, equivalently, Lagrangian relaxation would generate good lower bounds for the MINTB problem in practice.

References

1. Ahuja, R.K., Magnanti, T.L., Orlin, J.B.: Network Flows, Theory, Algorithm and Applications. Prentice Hall, Upper Saddle River (1993)
2. Bai, L., Hearn, D.W., Lawphongpanich, S.: Decomposition techniques for the minimum toll revenue problem. Networks **44**(2), 142–150 (2004)
3. Bai, L., Rubin, P.A.: Combinatorial benders cuts for the minimum tollbooth problem. Oper. Res. **57**(6), 1510–1522 (2009)
4. Bazaraa, M.S., Jarvis, J.J., Sherali, H.D.: Linear Programming and Network Flows. Wiley, Hoboken (1990)
5. Beckmann, M., McGuire, C., Winston, C.: Studies in the Economics of Transportation. Yale University Press, New Haven (1956)
6. Costa, M.C., Létocart, L., Roupin, F.: Minimal multicut and maximal integer multiflow: a survey. Eur. J. Oper. Res. **126**, 55–69 (2005)
7. CPLEX: CPLEX Optimization Inc., Incline Village (1996)
8. DeCorla-Souza, P.: Report to the transportation research board joint subcommittee on pricing. In: 82nd Annual Transportation Research Board Meeting, Washington (2005)
9. Florian, M., Guélat, J., Spiess, H.: An efficient implementation of the PARTAN variant of the linear approximation method for the network equilibrium problem. Networks **17**, 319–339 (1987)
10. Florian, M., Hearn, D.W.: Network equilibrium models and algorithms. In: Ball, M.O., Magnanti, T.L., Monma, C.L., Nemhauser, G.L. (eds.) Handbooks in Operations Research and Management Science, vol. 8: Network Routing, pp. 485–550. North-Holland, New York (1995)
11. GAMS: General Algebraic Modeling System. GAMS Development Corporation (1995)
12. Garg, N., Vazirani, V., Yannakakis, M.: Multiway cuts in directed and node weighted graphs. In: Proceedings for the 21st International Colloquium on Automata, Languages, and Programming, pp. 487–498, Jerusalem (1994)
13. Hearn, D.W., Lawphongpanich, S., Ventura, J.: Restricted simplicial decomposition: computation and xtensions. Math. Program. Study **31**, 99–118 (1995)
14. Hearn, D.W., Ramana, M.: Solving congestion toll pricing models. In: Marcotte, P., Nguyen, S. (eds.) Equilibrium and Advanced Transportation Modeling, pp. 109–124. Kluwer, Norwell (1998)
15. Kim, D., Pardalos, P.M.: A solution approach to the fixed charge network flow problem using a dynamic slope scaling procedure. Oper. Res. Lett. **24**(4), 195–203 (1999)
16. Lawphongpanich, S.: Dynamic slope scaling procedure and Lagrangian relaxation with subproblem approximation. J. Glob. Optim. **35**, 121–130 (2006)
17. LeBlanc, L.J., Morlok, E.K., Pierskalla, W.P.: An efficient approach to solving the road network equilibrium traffic assignment problem. Transp. Res. **9**, 309–318 (1975)
18. Mineta, N.Y.: National Strategy to Reduce Congestion on America's Transportation Network. U.S. Department of Transportation, Washington (2006)
19. Nahapetyan, A., Pardalos, P.M.: Adaptive dynamic cost updating procedure for solving fixed charge network flow problems. Comput. Optim. Appl. **39**(1), 37–50 (2008)
20. Pigou, A.C.: The Economics of Welfare. MacMillan, New York (1920)
21. Schrank, D., Lomax, T.: Urban Mobility Report 2007. Texas Transportation Institute, September 2007. http://mobility.tamu.edu. Accessed 1 May 2008
22. Todd, J.: Duke Student Math Aims to Alleviate Tollbooth Lines. Duke University News and Communications, June 2005. http://www.dukenews.duke.edu/2005/06/tollbooths.html. Accessed 1 May 2008
23. Verhoef, E.: Second-best congestion pricing in general networks: heuristic algorithms for finding second-best optimal toll levels and toll points. Transp. Res. B. **36**, 707–729 (2002)
24. Yang, H., Zhang, X.: Optimal toll design in second-best link-based congestion pricing. J. Transp. Res. Board Transp. Res. Rec. **1857**, 85–92 (2003)
25. Yildirim, M.B., Hearn, D.W. : A toll pricing framework for traffic assignment problems with elastic demand. In: Gendreau, M., Marcotte, P. (eds.) Current Trends in Transportation and Network Analysis: Papers in honor of Michael Florian, pp. 135–145. Kluwer, Norwell (2002)
26. Yildirim, M.B., Hearn, D.W.: A first best toll pricing framework for variable demand traffic assignment problems. Transp. Res. B. **39**, 659–678 (2005)
27. Zhang, X., Yang, H.: The optimal cordon-based network congestion pricing problem. Transp. Res. B. **38**, 517–537 (2004)