ON THE MINIMIZATION OF TRAFFIC CONGESTION IN ROAD NETWORKS WITH TOLLS

F. STEFANELLO, L.S. BURIOL, M.J. HIRSCH, P.M. PARDALOS, T. QUERIDO, M.G.C. RESENDE, AND M. RITT

Abstract. Population growth and the massive production of automotive vehicles have lead to the increase of traffic congestion problems. Traffic congestion today is not limited to large metropolitan areas, but is observed even in medium-sized cities and highways. Traffic engineering can contribute to lessen these problems. One possibility, explored in this paper, is to assign tolls to streets and roads, with the objective of inducing drivers to take alternative routes, and thus better distribute traffic across the road network. This assignment problem is often referred to as the *tollbooth problem* and it is *NP-hard*. In this paper, we propose mathematical formulations for two versions of the tollbooth problem that use piecewise-linear functions to approximate congestion cost. We also apply a previously proposed biased random-key genetic algorithm on a set of real-world instances, analyzing two ways of evaluating shortest paths. Experimental results show that the proposed piecewise-linear functions approximate the original convex function quite well and that the biased random-key genetic algorithm produces high-quality solutions.

1. Introduction

Transportation systems play an important role in modern life. Due to population growth and the massive production of vehicles, traffic congestion problems in metropolitan areas have become a common daily occurrence. To a commuter or traveler, congestion means loss of time, potentially missed business opportunities, and increased stress and frustration. To an employer, congestion means lost worker productivity, reduced trade opportunities, delivery delays, and increased costs (Wen, 2008). For example, a significant aspect is the value of wasted fuel and loss of productivity. In 2010, traffic congestion cost about US$115 billion in the 439 urban areas of the United States alone (Schrank et al., 2011).

Minimizing driving time directly impacts quality of life. One way to reduce travel time is by lowering congestion through the redistribution of traffic throughout the network. Improvements in transportation systems require a careful analysis of several factors. Different alternatives are evaluated using models that attempt to capture the nature of transportation systems and thus allow the estimation of the effect of future changes in system performance. Performance measures include efficiency in time and cost, security, and social and environmental impact, among others.

Date: January 21, 2013.
Key words and phrases. Combinatorial optimization, Transportation networks, Genetic algorithms, Tollbooth problem.
AT&T Labs Research Technical Report.

Several strategies have been proposed to reduce traffic congestion. Among them, the deployment of tolls on certain roads can induce drivers to choose alternative routes, thus reducing congestion as the result of better traffic flow distribution. Naturally, tolls can increase the cost of a trip, but this can be compensated with less travel time, reduced fuel cost, and lower amounts of stress. In the 1950s, Beckmann et al. (1956) proposed the use of tolls with this objective. This idea has made its way into modern transportation networks. In 1975, Singapore implemented a program called *Electronic Road Pricing* or ERP. Several cities in Europe and the United States, such as in London and San Diego, have begun to charge toll on their transportation networks (Bai et al., 2010). Tolls are also applied in some small European towns, like Peruggia (Italy), to reduce the number of people driving in downtown areas.

Determining the location of tollbooths[1] and their corresponding tariffs is a combinatorial optimization problem. This problem has aroused interest in the scientific community not only because of its intrinsic difficulty, but also because of the social importance and impact of its solution.

The optimization of transportation network performance has been widely discussed in the literature. The minimum tollbooth problem (MINTB), first introduced by Hearn and Ramana (1998), aims at minimizing the number of toll locations to achieve system optimality (MINSYS). Yang and Zhang (2003) formulate second-best link-based pricing as a bi-level program and solve it with a genetic algorithm. In Bai et al. (2010) it is shown that the problem is *NP-hard* and a local search heuristic was proposed. Another similar problem is to minimize total revenue (MINREV). MINREV is similar to MINSYS, but in this class of problems tolls can be negative as well as positive, while MINSYS does not accept negative tolls (Hearn and Ramana, 1998; Dial, 1999b;a; Hearn and Yildrim, 2002; Bai et al., 2004). For a complete review of the design and evaluation of road network pricing schemes we refer the reader to the survey by Tsekeris and Voß (2009).

Road and telecommunication routing problems have some similarities. They are both modeled as a directed weighted graph, where each link has capacity and delay (or link travel time), and a demand matrix defines the amount of flow required between each pair of nodes. In contrast to road networks, whose flows depend on the routes taken by the users, in telecommunication networks the flow is sent according to a protocol. One of the most commonly used protocols within autonomous systems is the *Open Shortest Path First* (OSPF) protocol that sends flow between origin and destination on a shortest path, splitting traffic evenly among alternative paths. A classical *NP-hard* optimization problem in this context is the weight setting problem (WSP) introduced by Fortz and Thorup (2004). The WSP assigns an integer weight to each link in a telecommunication network such that when flow is sent on a least-weight path, network congestion is minimized. Heuristics were successfully applied to solve the WSP (Fortz and Thorup, 2004; Ericsson et al., 2002; Buriol et al., 2005).

In this paper, we approach the tollbooth problem by routing on shortest paths as first studied in Buriol et al. (2010). The objective is to determine the location of a fixed number $\mathcal{K}$ of tollbooths and set their corresponding tariffs so that users travel on shortest paths between origin and destination, reducing network congestion. We

[1]We use the term *tollbooth* to refer to both traditional tollbooths as well as to sensors that read radio-frequency identification (RFID) tags from vehicles.

measure shortest paths in two ways. In the first way, we compute least cost paths based only on the tariffs of the tolled arcs. In the second, we calculate them based on toll tariffs and arc free flow times, where free flow time of an arc is defined to be the congestion-free time to traverse the arc. We also present a mathematical model for the minimum average link travel time and the tollbooth problem. We further propose two piecewise-linear functions that approximate an adapted convex travel cost function of the Bureau of Public Roads (1964) for measuring link congestion. Finally, we extend the work in Buriol et al. (2010) presenting a larger set of experiments.

This paper is organized as follows. In Section 2 we present mathematical models for the minimum average link travel time, the tollbooth problem, and two approximate piecewise-linear functions for travel cost. The biased randon-key genetic algorithm with local search proposed in Buriol et al. (2010) is presented in Section 3. Computational results are reported in Section 4. Finally, conclusions are drawn in Section 5.

2. Problem formulation

A road network can be represented as a directed graph $G = (V, A)$ where V represents the set of nodes (street or road intersections or points of interest), and A the set of arcs (street or road segments). Each arc $a \in A$ has an associated capacity c_a, and a time t_a, called the *free flow time*, necessary to transverse the unloaded arc a. To calculate the congestion on each link, a potential function Φ_a is computed as a function of the load or flow ℓ_a on arc a, along with p_a and β_a, two real-valued arc-tuning parameters. In addition, let

$$K = \{(o(1), d(1)), (o(2), d(2)), \dots, (o(|K|), d(|K|)\} \subseteq V \times V$$

denote the set of commodities or origin-destination (OD) pairs, where $o(k)$ and $d(k)$ represent, respectively, the origination and destination nodes for $k = 1, \dots, |K|$. Each commodity $k = 1, \dots, |K|$ has an associated demand of traffic flow $d_k = d_{o(k),d(k)}$, i.e., for each OD pair $(o(k), d(k))$, there is an associated flow d_k that emanates from node $o(k)$ and terminates in node $d(k)$. In this paper we address the problem in which all the demand is routed on the network, such that traffic congestion is minimized. To encourage traffic to take on particular routes, we resort to charging tolls on selected street or road segments.

Before we describe our mathematical models, some notation is introduced. We denote by $IN(v)$ the set of incoming arcs to node $v \in V$, by $OUT(v)$ the set of outgoing arcs from node $v \in V$, by $a = (a_t, a_h) \in A$ a directed arc of the network, where $a_t \in V$ and $a_h \in V$ are, respectively, the tail and head nodes of arc a, by $\mathcal{S} = \sum_{k=1}^{|K|} d_k$ the total sum of demands, and by $\mathcal{Q} \subseteq V$ the set of destination nodes. Moreover, we denote by Φ_a the traffic congestion of arc $a \in A$, and by $\mathcal{K}$ the number of tollbooths to deploy (tolls are levied on users of the network at tollbooths). The values of φ_a^u and φ_a^l are approximations of traffic congestion cost on arc $a \in A$ given by piecewise-linear functions. We note that throughout the paper we refer to flow and load interchangeably, as we do for commodity and demand.

In the next subsection we present a mathematical model of a relaxation of the tollbooth problem that does not take into account shortest paths. In Subsection 2.2 a complete model for the tollbooth problem is presented and in Subsection 2.3 we propose two piecewise-linear functions that approximate the convex cost function.

2.1. **Model for minimization of average user travel time (MM1).** The evaluation costs of a route can be defined in different ways according to specific goals. In this paper we use the potential function

$$\Phi = \textstyle\sum_{a\in A} \Phi_a, \text{ where } \Phi_a = \ell_a t_a \big[1 + \beta_a (\frac{\ell_a}{c_a})^{p_a}\big], \text{ for all } a \in A.$$

The average travel time is computed by simply normalizing Φ_a by $\mathcal{S}$. Function Φ_a is convex and nonlinear and is a strictly increasing function of l_a. A mathematical programming model of average user travel time is given in (1)–(5). Its goal is to determine flows on each arc such that network congestion is minimized. In this model, decision variables $x_a^q \in \mathbb{R}^+$ represent the total flow to destination $q \in \mathcal{Q}$ on arc $a \in A$, and variables $\ell_a \in \mathbb{R}^+$ represent the total flow on arc $a \in A$.

$$\min \Phi = \sum_{a\in A} \ell_a t_a \big[1 + \beta_a (\ell_a/c_a)^{p_a}\big]/\mathcal{S} \tag{1}$$

subject to:

$$\ell_a = \sum_{q\in\mathcal{Q}} x_a^q,\ \forall a \in A, \tag{2}$$

$$\sum_{a\in OUT(v)} x_a^q - \sum_{a\in IN(v)} x_a^q = d_{v,q},\ \forall v \in V\backslash\{q\},\ \forall q \in \mathcal{Q}, \tag{3}$$

$$x_a^q \geq 0, \forall a \in A,\ \forall q \in \mathcal{Q}, \tag{4}$$

$$\ell_a \geq 0,\ \forall a \in A. \tag{5}$$

Objective function (1) minimizes average user travel time. Constraints (2) define total flow on each arc $a \in A$ taking into consideration the contribution of all commodities. Constraints (3) guarantee flow conservation and (4)–(5) define the domains of the variables.

2.2. **Model for the tollbooth assignment problem (MM2).** A mathematical model for the tollbooth problem is given in (6)–(21). This model seeks to levy tolls on $\mathcal{K}$ arcs of the transportation network such that the average user travel time is minimized if traffic is routed on least-cost paths. Here, the cost of a path is defined to be the sum of the tolls levied on the arcs of the path. Note that MM2 differs from MM1 in that traffic in MM2 is routed on least-cost paths.

The decision variables for this model determine whether an arc will host a tollbooth and the amount of toll levied at each deployed tollbooth. Denote by $w_a \in \{0, P_l, P_l + 1, \dots, P_u\}$ as the toll tariff levied on arc $a \in A$, where $P_l \in \mathbb{N}^*$ and $P_u \in \mathbb{N}^+$ are the minimum and maximum values of tariffs, respectively. If no toll is levied on arc a, then $w_a = 0$. The binary decision variable $p_a = 1$ if and only if a tollbooth is deployed on arc $a \in A$. The auxiliary binary variable $y_a^q = 1$ if and only if arc $a \in A$ is part of a shortest path to destination node $q \in \mathcal{Q}$. Finally, auxiliary variable δ_v^q is the shortest-path distance from node $v \in V$ to destination node $q \in \mathcal{Q}$.

$$\min \ \Phi = \sum_{a\in A} \ell_a t_a \big[1+\beta_a(\ell_a/c_a)^{p_a}\big]/\mathcal{S} \tag{6}$$

Subject to:

$$\ell_a = \sum_{q\in\mathcal{Q}} x_a^q, \ \forall a \in A, \tag{7}$$

$$\sum_{a\in OUT(v)} x_a^q - \sum_{a\in IN(v)} x_a^q = d_{v,q}, \ \forall v \in V\backslash\{q\}, \ \forall q \in \mathcal{Q}, \tag{8}$$

$$C_a + w_a + \delta_{a_h}^q - \delta_{a_t}^q \geq 0, \ \forall a \in A, \ \forall q \in \mathcal{Q} \ , \tag{9}$$

$$\delta_q^q = 0, \ \forall q \in \mathcal{Q}, \tag{10}$$

$$C_a + w_a + \delta_{a_h}^q - \delta_{a_t}^q \geq (1-y_a^q)/M_1, \ \forall a \in A, \ \forall q \in \mathcal{Q}, \tag{11}$$

$$C_a + w_a + \delta_{a_h}^q - \delta_{a_t}^q \leq (1-y_a^q)M_2, \ \forall a \in A, \ \forall q \in \mathcal{Q}, \tag{12}$$

$$y_a^q \geq x_a^q, \ \forall a \in A, \ \forall q \in \mathcal{Q}, \tag{13}$$

$$M_3 y_a^q + M_3 y_b^q \leq 2M_3 - x_a^q + x_b^q, \ \forall a,b \in \mathsf{A}^2_{OUT(v)}, \forall v \in V, \ \forall q \in \mathcal{Q}, \tag{14}$$

$$P_l p_a \leq w_a \leq P_u p_a, \ \forall a \in A, \tag{15}$$

$$\sum_{a\in A} p_a = \mathcal{K}, \ \forall a \in A, \tag{16}$$

$$x_a^q \geq 0, \ \forall a \in A, \ \forall q \in \mathcal{Q}, \tag{17}$$

$$\ell_a \geq 0, \ \forall a \in A, \tag{18}$$

$$w_a \geq 0, \ \forall a \in A, \tag{19}$$

$$\delta_v^q \geq 0, \ \forall q \in \mathcal{Q}, \ \forall v \in V, \tag{20}$$

$$p_a \in \{0,1\}, \ \forall a \in A. \tag{21}$$

Objective function (6) minimizes average user travel time. Constraints (7) define the total flow on each arc $a \in A$ while constraints (8) impose flow conservation. Constraints (9) define the shortest path distance for each node $v \in V$ and each destination $q \in \mathcal{Q}$. For consistency, constraints (10) require, for all $q \in \mathcal{Q}$, that the shortest distance from q to itself be zero. Constraints (11) and (12) together with (9) and (10) determine whether arc $a \in A$ belongs to the shortest path and thus determine the values of y_a^q, for $q \in \mathcal{Q}$. Constraints (11) require that an arc that does not belong to the shortest path has reduced cost $C_a + w_a + \delta_{a_h}^q - \delta_{a_t}^q > 0$, where $C_a = \epsilon > 0$ is an sufficiently small term added to the arc cost so that when there are one or more zero-cost paths in the network, the flow is sent along those with least hop count. This also prevents flow from traversing zero-cost cycles. Constraints (12) assure that if the reduced cost of arc $a \in A$ and destination $q \in \mathcal{Q}$ is equal to zero, then arc a belongs to the shortest path to destination q, i.e. $y_a^q = 1$. Constraints (13) assure that flow is sent only on arcs belonging to a shortest path. Constraints (14) guarantee that flow is split evenly among all shortest paths. In these constraints, $\mathsf{A}^2_{OUT(v)}$ is the set of all ordered groups of two distinct elements of $OUT(v)$. We later discuss these constraints in more detail. Constraints (15) limit the minimum and maximum tariff for a deployed tollbooth. Constraints (16) require that exactly $\mathcal{K}$ tolls be deployed. The remaining constraints define the domains of the variables.

Constraints (14) come in pairs for each node $v \in V$. For every pair of outgoing links $a \in OUT(v)$ and $b \in OUT(v)$: $\{a,b\} \in \mathsf{A}^2_{OUT(v)}$ and $\{b,a\} \in \mathsf{A}^2_{OUT(v)}$, there are two corresponding constraints. They model load balancing by assuring that if the flow from node $v \in V$ to destination $q \in \mathcal{Q}$ is routed on both arcs $a \in A$ and $b \in A$, i.e. $y^q_a = y^q_b = 1$, then the flow on these arcs must be evenly split, i.e. $x^q_a = x^q_b$. To see this, suppose $y^q_a = y^q_b = 1$. The constraint for pair $\{a,b\} \in \mathsf{A}^2_{OUT(v)}$ implies that $x^q_a \leq x^q_b$. By symmetry the constraint for pair $\{b,a\} \in \mathsf{A}^2_{OUT(v)}$ implies that $x^q_a \geq x^q_b$. Consequently, $x^q_a = x^q_b$. Note that taking $M_3 \geq \max_{q \in \mathcal{Q}} \left(\sum_{v \in V} d_{v,q}\right)$ we assure that the right-hand-side of constraint (14) is bounded from below by M_3, making these constraints redundant for pairs of links with at most one of either y^k_a or y^k_b equal to one.

A model for OSPF routing, which also models shortest paths and even flow splitting, was previously proposed in Broström and Holmberg (2006). In their model a shortest path graph is built for each OD pair, while we opted for building a shortest path graph from all nodes to each node $q \in \mathcal{Q}$. This modification reduces the number of variables and constraints of the model.

We evaluate shortest paths in two ways. In the first approach, called SPT (Shortest Path Toll), we define the weight of an arc $a \in A$ to be the tariff w_a levied on that arc. In this case, we set $C_a = \epsilon$, an sufficiently small value. This way, when there are one or more zero-cost paths, the flow is always sent along paths having smallest hop count. In the second approach, called SPTF (Shortest Path Toll+Free flow time), we define the weight of an arc $a \in A$ to be the tariff w_a levied on the arc plus the free flow time t_a of the arc, i.e. parameter $C_a = t_a + \epsilon$. The value $\epsilon > 0$ is added to the cost with the same goal as in the case of SPT since it is possible that $t_a = 0$ for one or more arcs $a \in A$.

2.3. **Piecewise-linear functions for the models.**

The performance of commercial mixed integer linear programming solvers has improved considerably over the last few years. The two mathematical programming models presented so far have a nonlinear objective function Φ. To apply these solvers, one must first linearize Φ. In this subsection, we propose two piecewise-linear approximations of the function $\Phi = \sum_{a \in A} \Phi_a$.

The first linearization φ^u, is an overestimation, and under certain conditions is an upper bound of Φ. The second linearization φ^l is an underestimation and provides a lower bound of Φ. It is possible to apply these linearizations to any model with this type of non-linear function. We apply them to models MM1 and MM2.

Let Ω be the set of constraints (2)–(5) or (7)–(21) of the previously described mathematical models. On the one hand, for the case where Ω represents the constraints of the MM1 model, the approximation is called LMM1. On the other hand, when Ω represents the constraints of the MM2 model, we call the approximation LMM2.

In approximation φ^u, the cost function of each arc $a \in A$ is composed of a series of line segments sequentially connecting coordinates

$$(X_0, \Phi_a(X_0)), (X_1, \Phi_a(X_1)), \ldots, (X_n, \Phi_a(X_n)),$$

where values $X_0, X_1, \ldots, X_n$ are given such that $X_0 = 0$, and for $i = 1, \ldots, n$, $X_i \in \mathbb{R}$ and $X_i > X_{i-1}$.

If we denote the cost on arc $a \in A$ by φ_a^u, then the resulting mathematical programming model of the overestimation φ^u is given in (22)–(25).

$$\min \sum_{a\in A} \varphi_a^u \tag{22}$$

subject to:

$$\text{Constraints } \Omega \text{ are satisfied,} \tag{23}$$

$$(m_a^i/c_a)\ell_a + b_a^i \leq \varphi_a^u,\ \forall a \in A,\ \forall i = 1,\dots,n, \tag{24}$$

$$\varphi_a^u \geq 0,\ \forall a \in A, \tag{25}$$

where

$$m_a^i = (\Phi_a(X_i) - \Phi_a(X_{i-1}))/(X_i - X_{i-1}),$$
$$b_a^i = \Phi_a(X_i) - X_i m_a^i,$$

where

$$\Phi_a(X_i) = X_i c_a t_a (1 + \beta_a (X_i)^{p_a})/\mathcal{S}$$

for $X_0 = 0 < X_1 < \cdots < X_n$. Objective function (22) minimizes the approximation of average user travel time. Constraints (24) evaluate the partial cost on each arc by determining the approximate value φ_a^u for Φ_a according to load l_a. Constraints (25) define the domain of the variables.

The linearization requires the definition of the terms $X_0, X_1, \dots, X_n$ whose values are computed as a function of ℓ_a/c_a. The number of these terms can be arbitrarily defined according to the accuracy required for the linearization of the cost function, or according to characteristics of the set of instances. This linearization requires a balance between the accuracy of the computed solution and the time to compute the linearization. With a large number n, the linearization tends to provide a better approximation of the original value, while a small value of n can reduce computational times since each element entails $|A|$ additional constraints in the model.

A second linearization, which we denote by φ_a^l, is an underestimation and gives us a lower bound on Φ_a. The mathematical model of this linearization is very similar to that of the overestimation. However, to estimate φ_a^l, we first compute the slope $m_a(x)$ of Φ_a at $x = (X_{i-1} + X_i)/2$, for $i = 1, \dots, n$, as

$$m_a(x) = \frac{\partial \Phi_a}{\partial x} = \frac{t_a}{\mathcal{S}} + \frac{(p_a+1)t_a\beta_a x^{p_a}}{c_a^{p_a}\mathcal{S}}.$$

Given x and $m_a(x)$, the independent term can be easily computed.

The linearizations φ_a^l and φ_a^u produce, respectively, an underestimation and an overestimation of Φ_a, as Proposition 1 states.

Proposition 1. *Let $\varphi^u = \sum_{a\in A} \varphi_a^u$, $\varphi^l = \sum_{a\in A} \varphi_a^l$, and as before $\Phi = \sum_{a\in A} \Phi_a$. Let $X_0, X_1, \dots, X_n$ be the values for which the appoximation is computed. If $\ell_a/c_a \leq X_n, \forall a \in A$, then $\varphi^l \leq \Phi \leq \varphi^u$.*

Proof. By construction $\varphi_a^l \leq \Phi_a$, then $\Phi = \sum_{a\in A} \Phi_a \geq \sum_{a\in A} \varphi_a^l = \varphi^l$. Thus $\Phi \geq \varphi^l$. Furthermore, if $\ell_a/c_a \leq X_n$, then by construction $\varphi_a^u \geq \Phi_a$, which implies that $\varphi^u = \sum_{a\in A} \varphi_a^u \geq \sum_{a\in A} \Phi_a = \Phi$. Thus $\varphi^u \geq \Phi$. Therefore $\varphi^l \leq \Phi \leq \varphi^u$. □

Note that for Propostion 1 to hold we do not make the assumption that the underestimation φ^l be a lower bound of Φ, while the overestimation φ^u requires that $\ell_a/c_a \leq X_n, \forall a \in A$ be true for the propostion to hold.

A representation of the functions φ_a^u, φ_a^l, and Φ is depicted in Figure 1. It shows the cost function Φ (solid line) as well as the linear piecewise-linear cost functions φ^u and φ^l for an arc $a \in A$ with $t_a = 5$, $c_a = 200$, $p_a = 4$, $\beta_a = 0.15$, and $\mathcal{S} = 1000$ using with $\{X_0, X_1, \ldots, X_6\} = \{0, 0.65, 1, 1.25, 1.7, 2.7, 5\}$. Observe that there is a higher concentration of points X in the range $\frac{l_a}{c_a} = [0.65; 1.25]$. This is so because the flow on a large number of arcs is concentrated around the capacity of the arc. Thus to get a good approximation requires that many X values be situated around $\frac{l_a}{c_a} = 1$. Note that a value of $\frac{l_a}{c_a} > 1$ indicates that the arc is overloaded.

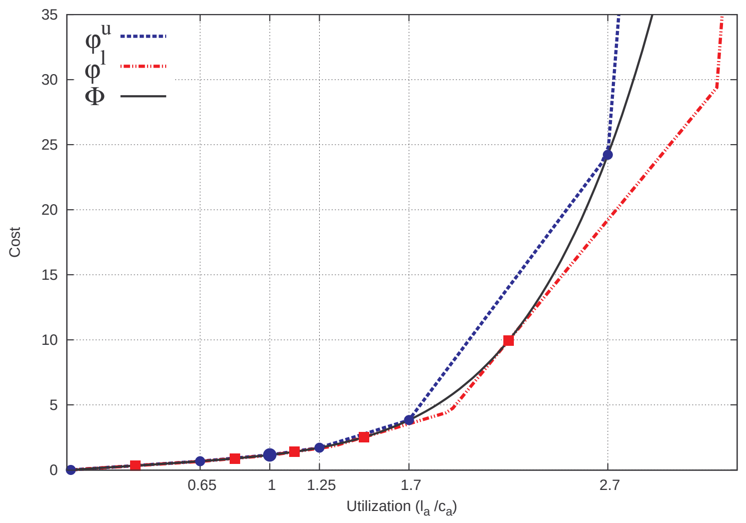

FIGURE 1. Comparison of the cost function with the linear piecewise-linear cost function.

3. A BIASED RANDOM-KEY GENETIC ALGORITHM

In this section we describe the biased random-key genetic algorithm (BRKGA) for the tollbooth problem, proposed in Buriol et al. (2010).

A random-key genetic algorithm (RKGA) is a metaheuristic, originally proposed by Bean (1994), for finding optimal or near-optimal solutions to optimization problems. RKGAs encode solutions as vectors of random keys, i.e. randomly generated real numbers in the interval $(0, 1]$. A RKGA starts with a set (or *population*) of p random vectors of size n. Parameter n depends on the encoding while parameter p is user-defined. Starting from the initial population, the algorithm generates a series of populations. Each iteration of the algorithm is called a *generation*. The algorithm evolves the population over the generations by combining pairs of solutions from one generation to produce offspring solutions of the following generation.

RKGAs rely on *decoders* to translate a vector of random keys into a solution of the optimization problem being solved. A decoder is deterministic algorithm that takes as input a vector of random keys and returns a feasible solution of the optimization problem as well as its cost (or *fitness*).

At the k-th generation, the decoder is applied to all newly created random keys and the population is partitioned into a smaller set of p_e elite solutions, i.e., the best fittest p_e solutions in the population and another larger set of $p - p_e > p_e$ non-elite solutions. Population $k+1$ is generated as follows. All p_e elite solutions of population k are copied without change to population $k+1$. This elitist strategy maintains the best solution on hand. In biology, as well as in genetic algorithms, evolution only occurs if mutation is present. As opposed to most genetic algorithms, RKGAs do not use a mutation operator, where each component of the solutions is modified with small probability. Instead p_m *mutants* are added to population $k+1$. A mutant is simply a vector of random keys, generated in the same way a solution of the initial population is generated.

With $p_e + p_m$ solutions accounted for in population $k+1$, $p - p_e - p_m$ additional solutions must be generated to complete the p solutions that make up population $k+1$. This is done through *mating* or *crossover*. In the RKGA of Bean (1994), two solutions are selected at random from the entire population. One is parent-A while the other is parent-B. A child C is produced by combining the parents using parameterized uniform crossover (Spears and DeJong, 1991). Let $\rho_A > 1/2$ be the probability that the offspring solution inherits the key of parent-A and $\rho_B = 1 - \rho_A$ be the probability that it inherits the key of parent-B, i.e.

$$c_i = \begin{cases} a_i & \text{with probability } \rho_A, \\ b_i & \text{with probability } \rho_B = 1 - \rho_A, \end{cases}$$

where a_i and b_i are, respectively, the i-th key of parent-A and parent-B, for $i = 1, \ldots, n$. This crossover always produces a feasible solution since c is also a vector of random keys and by definition the decoder takes as input any vector of random keys and outputs a feasible solution.

Biased random-key genetic algorithms (Gonçalves and Resende, 2011) differ from Bean's algorithm in the way parents are selected. In a BRKGA parent-A is always selected at random from the set of p_e elite solutions while parent-B is selected at random from the set of $p - p_e$ non-elite solutions. The selection process is *biased* since an elite solution s has probability $Pr(s) = 1/p_e$ of being selected for mating while a non-elite solution $\bar{s}$ is selected with probability $Pr(\bar{s}) = 1/(p - p_e)$. Since $p - p_e > p_e$, then $Pr(s) > Pr(\bar{s})$. In addition, elite solutions have a higher probability of passing on their random keys since probability $\rho_A > 1/2$. Though the difference between RKGAs and BRKGAs is small, the resulting heuristics behave quite differently. Experimental results in Gonçalves et al. (2012) show that BRKGAs are almost always faster and more effective than RKGAs.

To describe a BRKGA, one need only show how solutions are encoded and decoded, what choice of parameters p, p_e, p_m, and ρ_A were made, and how the algorithm stops. We describe encoding and decoding next and give values for parameters as well as the stopping criterion in Section 4.

Solutions are encoded as a $2 \times |A|$ vector $\mathcal{X}$ of random keys, where $|A|$ is the cardinality of the set A of arcs in the network. The first $|A|$ keys correspond to toll tariffs while the last $|A|$ keys are used to locate the $\mathcal{K}$ tolls.

The decoder has two phases. In the first phase tolls are selected and arc tariffs are set directly from the random keys. In the second phase, a local improvement procedure attempts to change the tariffs with the goal of reducing the value of the objective function. Each tolled arc a has a tariff in the interval $[1, w_{\max}]$, where $w_{\max}$ is an input parameter. The tariff for arc a is simply decoded as $\lceil \mathcal{X}_a \times w_{\max} \rceil$ if it is tolled or zero otherwise. To determine the $\mathcal{K}$ tolled arcs we simply select as tolled the arcs corresponding to the $\mathcal{K}$ largest keys among $\mathcal{X}_{|A|+1}, \mathcal{X}_{|A|+2}, \ldots, \mathcal{X}_{2\times|A|}$. Let $a'_1, a'_2, \ldots, a'_{\mathcal{K}}$ be the $\mathcal{K}$ tolled arcs, then

$$\mathcal{X}_{a'_1} \geq \mathcal{X}_{a'_2} \geq \cdots \geq \mathcal{X}_{a'_{\mathcal{K}}} \geq \mathcal{X}_{a_j}, \text{ for all } a_j \notin \{a'_1, a'_2, \ldots, a'_{\mathcal{K}}\}.$$

Ties are broken by the arc index.

Demands are routed forward to their destinations on shortest weight paths. For SPT, tolled links have weights equal to their tariffs and untolled links are assumed to have weight zero. For SPFT, we add to the tariff the free flow time to define the weight of all tolled arcs, while each untolled arc has weight equal to its free flow time. Depending on the number of tolls and the network, there can be several shortest paths of cost zero (especially in the case SPT). In this case, we use the path with the least number of hops. Traffic at intermediate nodes is split equally among all outgoing links on shortest paths to the destination. After the flow is defined, the fitness of the solution is computed by evaluating the objective function Φ.

The second phase of the decoder is local improvement. Local search is applied to the solution produced in the first phase of the decoder. In short, it works as follows. Let $A^* \subseteq A$ be the $q = \min\{|A|, 5\}$ arcs having the largest congestion costs Φ_a, i.e. $|A^*| = q$ and $\Phi_{a^*} \geq \Phi_a$, for all pairs $\{a^*, a\}$ such that $a^* \in A^*$ and $a \in A \setminus A^*$. For each arc $a^* \in A^*$, in case it is tolled, its weight is increased by one unit at a time, to induce a reduction of its load. The unit-increase is repeated until either the weight reaches w_{max} or Φ no longer improves. If no improvement in the objective function is achieved, the weight is reset to its initial value. In case the arc is not currently tolled, a new toll is installed on the arc with initial weight one, and a toll is removed from some other link tested in circular order. If no reduction in the objective function is achieved, the solution is reversed to its original state. Every time a reduction in Φ is achieved, a new set A^* is computed and the local search restarts. The procedure stops at a local minimum when there is no improved solution changing the weights of the candidate arcs in set A^*.

4. Computational results

In this section we present computational experiments with the models and algorithms introduced in the previous sections of this paper. Initially, we describe the dataset used in the experiments. Then, we detail three sets of experiments. The first set evaluates the mathematical models MM1 and LMM1. The second set of experiments evaluates the full model with piecewise linear function MM2, which considers the shortest paths constraints with even split of loads. The last set of experiments evaluates the biased random-key genetic algorithm presented in Section 3.

The experiments were done on a computer with an Intel Core i7 930 processor running at 2.80 GHz, with 12 GB of DDR3 RAM of main memory, and Ubuntu 10.04 Linux operating system. The biased random-key genetic algorithms (BRKGA) were implemented in C and compiled with the `gcc` compiler, version 4.4.3, with optimization flag `-O3`. The experiments with the BRKGA were done

Table 1. Attributes for the instances are given in each row. For each instance, its row lists the set identification (*S1* or *S2*), instance name, number of vertices, links, origin-destination pairs, number of vertices in which traffic originates (Source nodes), and number of nodes in which traffic terminates (Sink nodes).

Set	**Instance**	**Vertices**	**Links**	**OD pairs**	**Source nodes**	**Sink nodes**
	`SiouxFalls_08`	8	16	48	8	8
	`SiouxFalls_09`	9	26	68	9	9
	`SiouxFalls_10`	10	36	84	10	10
S1	`SiouxFalls_12`	12	38	126	12	12
	`SiouxFalls_14`	14	36	172	14	14
	`SiouxFalls_16`	16	50	218	16	16
	`SiouxFalls`	24	76	528	24	24
	`Friedrichshain Center`	223	514	506	23	23
	`Prenzlauerberg Center`	350	717	1406	38	38
	`Tiergarten Center`	361	749	644	26	26
	`Mitte Center`	398	857	1260	36	36
S2	`Anaheim`	416	914	1406	38	38
	`MPF Center`	974	2153	9505	98	98
	`Barcelona`	1020	2522	7922	97	108
	`Winnipeg`	1052	2836	4345	135	138
	`ChicagoSketch`	933	2950	9351	386	386

with a population size $p = 50$, an elite set of size $p_e = .25p$, a mutant set of size $p_m = .05p$, and an elite key inheritance probability of $\rho_A = .7$. The commercial solver CPLEX 12.3[2] was used to solve the proposed linearizations of the mathematical linear models, while MOSEK[3] was used to solve the mathematical model MM1 (with convex objective function).

Table 1 details six synthetic instances (*S1*) and eleven real-world instances (*S2*) considered in our experiments and made available by Bar-Gera (2012).

To test model LMM2, we created the instances in set *S1* from instance `SiouxFalls` of *S2* by removing from `SiouxFalls` some of its nodes and their adjacent links as well as all origin-destination pairs where these nodes are either origin or destination nodes. Let $n < |V|$ be the new number of nodes. We choose to remove nodes

$$v \in V : v = \left\lfloor k\frac{|V|}{|V|-n} + 1 \right\rfloor \text{ with } k = 0, \dots, |V| - n - 1.$$

Let $v, a, b \in V$ be nodes such that $a \in OUT(v)$ and $b \in IN(v)$. Furthermore, let a_t (b_t) and a_h (b_h) be, respectively, the tail and head nodes of links a (b). We create a link a' from a_t to b_t if there does not already exist a link between a_h and b_t and furthermore $|OUT(b_t)| < 4$ or $|IN(a_h)| < 4$. Link a' has the same characteristics (free flow time, capacity, etc.) of link a. After all extensions, we remove from the network all arcs $a \in OUT(v) \cup IN(v)$ as well as node v.

4.1. **Results for models MM1 and LMM1.** The first set of experiments evaluates the models when solved with commercial solvers. Table 2 presents, for each instance, the objective functions Φ, and the lower and upper bounds φ^l and φ^u, respectively.

[2]`www-01.ibm.com/software/integration/optimization/cplex-optimizer`
[3]`www.mosek.com`

In the first two columns after the instance names, we present the objective function value Φ and the computational time for model MM1 obtained with the nonlinear solver MOSEK 6.0 using the modeling system GAMS[4]. A few non-linear solvers are part of the GAMS system and we evaluated the performance of all of them. Some of them are general non-linear solvers, and have no specific routines for convex functions. Most were not able to solve the larger instances. MOSEK presented the best performance in terms of running time and for this reason we report only the results obtained with MOSEK. The next columns present results for CPLEX 12.3 with the proposed piecewise linear functions φ^l and φ^u, respectively the lower and upper estimations of function Φ. In each case, we show the objective function values in columns φ^l and φ^u, as well as $\Phi\{\varphi^l\}$ and $\Phi\{\varphi^u\}$, the values of Φ considering the arc loads obtained by the different approximations. The computational times are reported in seconds.

From the results in Table 2, three main observations that can be made. First, there are small gaps between φ^l and $\Phi\{\varphi^l\}$, as well as between φ^u and $\Phi\{\varphi^u\}$, i.e. both piecewise linear functions have values slightly close to the corresponding evaluation Φ. In a small number of cases the gap is significant and we observe that, as expected, this occurs in instances with higher average or higher maximum utilization (ℓ_a/c_a), like `Barcelona` and `Winnipeg`. Second, we compare the results for models MM1 and LMM1. The gaps between φ^l and Φ, and between φ^u and Φ, are also small, which means that the piecewise functions have similar values to the original convex function Φ. However, for most of the instances, the computational time spent by MOSEK on the convex function is 2-to-4 orders of magnitude greater than the time spent by CPLEX on the piecewise linear functions. The only case where solving the model with a piecewise linear function (case of φ^u with CPLEX) took longer than solving the model with the convex function Φ (using MOSEK) was for instance `ChicagoSketch`. However, CPLEX found good solutions quickly, and spent most of the time certifying optimality. For example, CPLEX found solutions with a gap of 3% with respect to the optimal solution in about 650 seconds, while MOSEK needed more than 1600 seconds to reach this gap.

The last important observation is that the MM1 model is a relaxation of MM2. Moreover, the shortest paths and even-load constraints of model MM2 add a considerable number of variables and constraints to the model. Thus, evaluating MM2 with a convex function became impracticable in terms of computational time, and for this reason no corresponding results are reported. In the next experiment we evaluate both approximations models for the full model (MM2).

4.2. **Results for the Tollbooth Problem with linear piecewise cost (LMM2).** This set of experiments tests the performance of CPLEX on MM2, the model that includes shortest paths and even-split constraints, with the two piecewise linear functions introduced in Section 2.3. Using both schemes to evaluate shortest path (SPT and SPTF), we run the model with both piecewise linear functions.

Tables 3 and 4 present results for model LMM2 when the shortest path is calculated considering only the toll tariffs (SPT), and for tariffs plus the free flow time (SPTF), respectively. For each instance, we tested several scenarios of $\mathcal{K}$. For each scenario we present the objective function values of approximations φ^l and φ^u obtained by CPLEX, the corresponding $\Phi\{\varphi^l\}$ and $\Phi\{\varphi^u\}$ values (as described in

[4]`www.gams.com`

TABLE 2. Computational results for MM1 and LMM1.

	MM1		**LMM1-lower estimate**			**LMM1-upper estimate**		
Instance	$\mathbf{\Phi}$	**Time(s)**	$\varphi^{\mathbf{l}}$	$\mathbf{\Phi}\{\varphi^{\mathbf{l}}\}$	**Time(s)**	$\varphi^{\mathbf{u}}$	$\mathbf{\Phi}\{\varphi^{\mathbf{u}}\}$	**Time(s)**
`SiouxFalls_08`	8.77	0.0	8.69	8.77	0.0	8.97	8.77	0.0
`SiouxFalls_09`	6.32	0.0	6.26	6.32	0.0	6.46	6.32	0.0
`SiouxFalls_10`	6.71	0.0	6.62	6.72	0.0	6.81	6.71	0.0
`SiouxFalls_12`	11.46	0.0	10.92	11.47	0.0	12.69	11.72	0.0
`SiouxFalls_14`	64.69	0.0	45.87	64.79	0.0	119.34	64.79	0.0
`SiouxFalls_16`	10.11	0.0	9.97	10.15	0.0	10.33	10.18	0.0
`SiouxFalls_18`	10.70	0.0	10.37	10.93	0.0	11.16	10.87	0.0
`SiouxFalls`	19.95	0.1	18.10	20.77	0.1	21.68	20.52	0.1
`Friedrichshain Center`	42.47	7.7	39.57	43.34	0.0	47.61	43.11	0.1
`Prenzlauerberg Center`	59.90	70.3	56.98	61.07	0.1	67.21	60.81	0.1
`Tiergarten Center`	52.57	164.5	47.63	53.63	0.1	59.68	52.91	0.1
`Mitte Center`	62.36	30.8	58.76	63.59	0.2	71.08	63.11	0.3
`Anaheim`	12.46	204.8	12.26	12.49	0.5	12.91	12.48	0.5
`MPF Center`	65.88	1,988.0	60.94	67.63	2.8	75.11	66.57	3.7
`Barcelona`	6.87	6,174.8	6.54	11.75	1.9	6.54	11.75	1.9
`Winnipeg`	13.67	2,189.0	12.25	20.83	4.7	12.25	20.83	4.7
`ChicagoSketch`	14.24	2,004.9	14.09	14.31	1,154.6	14.50	14.30	2,465.1

the previous subsection), the gap returned by the solver for a time limit of 1800 seconds, and finally the running times in seconds. The null values (-) indicate that a feasible solution was not found within the time limit.

Tables 3 and 4 illustrate the difficulty in solving these instances with CPLEX. For most of the instances no optimal solution was found within 30 minutes, and for many of them even a feasible solution was not found in this time limit. A small increase in the instance size implies in a large increase in the computational effort spent to solve it. We observe that for SPT the solver has more difficulty in finding an initial solution, and the gap returned by the solver is slightly higher in comparison with the SPTF path calculation. Furthermore, the computational time is slightly reduced for SPTF, and φ^l was computed slightly faster than was φ^u.

Results for instances `SiouxFalls_06` and `SiouxFalls_08` were found for φ^l and φ^u in almost no time, and for this reason they were omitted from the table. On the other hand, results were omitted for `SiouxFalls_16`, for both piecewise linear functions and SPT and SPTF path calculations since the solver was not able to find any feasible solution within the time limit.

In summary, Table 2 shows that solving the simplified model MM1, that is MM2 without the shortest paths computation and even-load constraints, takes a considerable time, while their linearized versions φ^l and φ^u are calculated very quickly in almost all cases. Tables 3 and 4, on the other hand, show that the linearizations φ^l and φ^u of the full model MM2 takes a long time even for small instances. Thus, these results motivated us to propose an heuristic solution to solve the tollbooth problem, and the results of the proposed biased random-key genetic algorithm are presented in the next subsection.

4.3. **Results for the biased random-key genetic algorithm.** This section presents results for the biased random-key genetic algorithm on the ten instances of class *S2*. We extend the experimental study performed by Buriol et al. (2010) which presents results for three of these instances (the other seven were only made available recently). Moreover, we provide an analysis of the best solution for each combination of instance and $\mathcal{K}$.

In the first experiment we compare the results obtained between the genetic algorithm with the simple decoder (BRKGA) and the genetic algorithm with the decoder with local search (BRKGA+LS). First, we run the BRKGA+LS with time limit of 3600 seconds, (except for large instance `ChicagoSketch` for which we ran with a time limit of 7200 seconds). We allow a maximum number of generations to be 2000, and the maximum number of generations without improvement to be 100 (5% of maximum number of generations). Later, we ran the BRKGA procedure with the same running times obtained in the previous experiment without other stopping criteria.

Table 5 shows the results obtained, averaged over five runs with different random seeds and over seven different values of $\mathcal{K}$ (the number of tollbooths used in this experiment are the same as shown in Table 8). In this table we present the average number of generations (Gens) and the objective value(Φ) for each instance and how the shortest path was evaluated. Finally, the last column presents the average running times in seconds of both algorithms.

This table shows that BRKGA+LS requires more computational effort per generation than BRKGA. However, it produces better results if the same amount of time is given. Columns Φ of the algorithms show that BRKGA found a better

TABLE 3. Computational results for LMM2 to SPT.

		Approximation		**Objective Function**		**Solver *gap***		**Time(s)**	
Instance	$\mathcal{K}$	$\varphi^{\mathbf{l}}$	$\varphi^{\mathbf{u}}$	$\mathbf{\Phi}\{\varphi^{\mathbf{l}}\}$	$\mathbf{\Phi}\{\varphi^{\mathbf{l}}\}$	$\varphi^{\mathbf{l}}$	$\varphi^{\mathbf{u}}$	$\varphi^{\mathbf{l}}$	$\varphi^{\mathbf{u}}$
	2	6.47	6.70	6.52	6.52	0.00	0.00	91.7	8.1
	5	6.35	6.58	6.38	6.38	0.00	0.00	12.4	94.4
	7	6.31	6.54	6.37	6.40	0.00	0.00	26.3	33.6
`SiouxFalls_09`	10	6.27	6.47	6.33	6.33	0.00	0.00	4.1	35.2
	13	6.27	6.46	6.32	6.32	0.00	0.00	3.7	0.6
	15	6.27	6.46	6.32	6.32	0.00	0.00	1.3	15.3
	18	6.27	6.46	6.32	6.32	0.00	0.00	0.8	1,581.7
	20	6.27	6.46	6.32	6.32	0.00	0.00	0.4	3.5
	3	-	-	-	-	-	-	1,800.0	1,800.0
	7	-	-	-	-	-	-	1,800.0	1,800.0
	10	-	7.36	-	7.22	-	6.92	1,800.0	1,800.0
`SiouxFalls_10`	14	-	-	-	-	-	-	1,800.0	1,800.0
	18	6.75	-	6.81	-	1.48	-	1,800.0	1,800.0
	21	6.70	6.90	6.76	6.78	0.59	0.42	1,800.0	1,800.0
	25	6.70	-	6.76	-	0.50	-	1,800.0	1,800.0
	28	6.70	6.90	6.76	6.78	0.23	0.21	1,800.0	1,800.0
	3	-	-	-	-	-	-	1,800.0	1,800.0
	7	-	-	-	-	-	-	1,800.0	1,800.0
	11	-	-	-	-	-	-	1,800.0	1,800.0
`SiouxFalls_12`	15	-	-	-	-	-	-	1,800.0	1,800.0
	19	-	-	-	-	-	-	1,800.0	1,800.0
	22	-	-	-	-	-	-	1,800.0	1,800.0
	26	-	-	-	-	-	-	1,800.0	1,800.0
	30	-	-	-	-	-	-	1,800.0	1,800.0
	3	46.72	120.69	65.69	65.69	1.63	0.94	1,800.0	1,800.0
	7	46.24	120.08	65.13	65.13	0.75	0.60	1,800.0	1,800.0
	10	46.15	119.69	64.97	64.97	0.55	0.27	1,800.0	1,800.0
`SiouxFalls_14`	14	-	-	-	-	-	-	1,800.0	1,800.0
	18	-	119.65	-	64.99	-	0.21	1,800.0	1,800.0
	21	46.14	-	64.96	-	0.49	-	1,800.0	1,800.0
	25	-	119.65	-	64.99	-	0.22	1,800.0	1,800.0
	28	46.14	-	64.96	-	0.39	-	1,800.0	1,800.0

Table 4. Computational results for LMM2 to SPTF.

		Approximation		**Objective Function**		**Solver *gap***		**Time(s)**	
Instance	$\mathcal{K}$	φ^{l}	φ^{u}	$\mathbf{\Phi}\{\varphi^{\mathrm{l}}\}$	$\mathbf{\Phi}\{\varphi^{\mathrm{l}}\}$	φ^{l}	φ^{u}	φ^{l}	φ^{u}
	2	6.28	6.47	6.33	6.33	0.00	0.00	0.5	0.3
	5	6.27	6.46	6.32	6.32	0.00	0.00	0.4	0.4
	7	6.27	6.46	6.32	6.32	0.00	0.00	0.3	9.2
	10	6.27	6.46	6.32	6.32	0.00	0.00	0.2	573.3
SiouxFalls_09	13	6.27	6.46	6.32	6.32	0.00	0.00	0.3	1.1
	15	6.27	6.46	6.32	6.32	0.00	0.00	0.4	0.4
	18	6.27	6.46	6.32	6.32	0.00	0.00	0.4	0.3
	20	6.27	6.46	6.32	6.32	0.00	0.00	0.4	0.4
	3	6.73	6.93	6.79	6.79	0.00	0.00	92.7	176.4
	7	6.70	6.90	6.76	6.78	0.00	0.15	203.2	1,800.0
	10	6.70	6.90	6.76	6.78	0.57	0.48	1,800.0	1,800.0
	14	-	6.90	-	6.78	-	0.31	1,800.0	1,800.0
SiouxFalls_10	18	6.70	6.90	6.76	6.78	0.56	0.51	1,800.0	1,800.0
	21	6.70	6.90	6.76	6.78	0.32	0.51	1,800.0	1,800.0
	25	6.70	-	6.76	-	0.36	-	1,800.0	1,800.0
	28	6.70	6.90	6.76	6.78	0.04	0.51	1,800.0	1,800.0
	3	11.19	13.09	11.67	11.82	0.00	1.87	1,748.4	1,800.0
	7	11.18	13.05	11.66	11.83	0.39	2.44	1,800.0	1,800.0
	11	-	-	-	-	-	-	1,800.0	1,800.0
	15	-	-	-	-	-	-	1,800.0	1,800.0
SiouxFalls_12	19	-	-	-	-	-	-	1,800.0	1,800.0
	22	-	-	-	-	-	-	1,800.0	1,800.0
	26	-	13.02	-	11.80	-	1.25	1,800.0	1,800.0
	30	11.18	-	11.66	-	1.64	-	1,800.0	1,800.0
	3	46.24	119.73	65.09	65.09	0.00	0.00	1,019.3	133.3
	7	46.14	119.65	64.96	64.99	0.48	0.25	1,800.0	1,800.0
	10	-	119.65	-	64.99	-	0.18	1,800.0	1,800.0
	14	-	-	-	-	-	-	1,800.0	1,800.0
SiouxFalls_14	18	46.14	119.65	64.96	64.99	0.36	0.17	1,800.0	1,800.0
	21	-	-	-	-	-	-	1,800.0	1,800.0
	25	-	119.65	-	64.99	-	0.21	1,800.0	1,800.0
	28	-	119.65	-	64.99	-	0.13	1,800.0	1,800.0

Table 5. Comparison of BRKGA and BRKGA+LS algorithms.

		BRKGA		**BRKGA+LS**		
Type	**Instance**	**Gens**	**Φ**	**Gens**	**Φ**	**Time(s)**
SPT	`SiouxFalls`	1,837.37	51.04	343.00	**30.71**	18.20
	`Friedrichshain Center`	2,232.63	49.66	769.80	**47.32**	231.08
	`Prenzlauerberg Center`	2,441.14	69.06	997.86	**67.91**	607.86
	`Tiergarten Center`	2,301.60	56.59	810.29	**55.84**	415.44
	`Mitte Center`	2,845.14	70.39	981.83	**69.60**	815.14
	`Anaheim`	4,175.57	14.12	1,447.03	**13.82**	1,369.21
	`MPF Center`	1,599.06	**80.65**	484.14	81.20	3,345.36
	`Barcelona`	1,391.54	16.47	419.03	**11.21**	3,361.03
	`Winnipeg`	1,055.29	35.02	326.71	**24.21**	3,457.38
	`ChicagoSketch`	862.06	194.27	254.34	**56.10**	7,209.72
SPTF	`SiouxFalls`	1,345.80	26.96	326.37	**22.83**	13.93
	`Friedrichshain Center`	759.89	45.57	352.57	**45.26**	82.17
	`Prenzlauerberg Center`	881.71	67.17	452.60	**66.22**	228.80
	`Tiergarten Center`	719.89	53.30	357.03	**53.56**	136.14
	`Mitte Center`	916.83	66.24	460.06	**65.65**	275.19
	`Anaheim`	1,593.29	13.15	714.77	**13.10**	556.12
	`MPF Center`	1,060.06	**69.19**	551.60	69.66	2,338.86
	`Barcelona`	1,222.00	**9.30**	510.86	9.31	3,109.35
	`Winnipeg`	947.46	22.41	432.03	**19.67**	3,309.19
	`ChicagoSketch`	844.34	21.97	335.97	**18.63**	7,208.79

average objective value only in 3 of 20 cases, demonstrating that the local search procedure is able to improve solution quality.

Another interesting observation is that BRKGA and BRKGA+LS spend less computational time when shortest paths are being calculated as SPTF, in comparison with SPT. This occurs because the number of alternative paths increases when we consider only tolls in the computation of the shortest paths, thus resulting in more time to compute and update shortest paths and route demands. This increase in the number of alternative shortest paths is expected since in the SPT case there are many arcs with zero cost (with no toll installed). Furthermore, arc weights having small integer values make it more likely that alternative paths occur.

Since BRKGA+LS was shown to outperform BRKGA, the remaining analysis in this subsection is limited to BRKGA+LS. Table 8 shows averages over five runs of BRKGA+LS and a comparison between SPT and SPTF. For each $\mathcal{K}$ (number of installed tolls), we show the number of generations (Gens), best solution value (Best Φ) over the five runs, average fitness values (Avg Φ), standard deviation (SD), and average running times in seconds.

The first observation is that when $\mathcal{K}$ increases the Φ values tend to decrease and present a smaller variance. Moreover, in most cases, the best solution values were not found with larger $\mathcal{K}$, which is close to $|E|$ (the number of arcs), but rather for $\mathcal{K} \approx \frac{|E|}{2}$, or slightly larger values. With small $\mathcal{K}$ the tolled arcs tend do not have flow, which impairs network traffic. On the other hand, with a large $\mathcal{K}$ the search space increases considerable, making the problem harder to solve. Also, we can see that the standard deviation is small in most cases showing that the algorithm is robust. Another implication of having a small $\mathcal{K}$ is that the values of Φ tend to be better than for SPTF for the same aforementioned reason.

Table 6. Detailed results of SPT and SPTF for the BRKGA+LS algorithm.

		SPT					SPTF				
Instance	$\mathcal{K}$	**Gens**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**	**Gens**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**
	10	688.10	50.51	59.05	7.71	11.90	496.10	25.19	28.09	3.06	9.46
	20	967.50	34.12	60.67	27.53	16.89	713.90	22.53	26.11	3.25	12.90
	30	1,474.00	30.69	54.97	24.82	25.23	1,062.30	22.13	25.06	2.60	18.20
`SiouxFalls`	40	1,277.70	22.62	39.18	17.07	21.64	1,016.10	21.97	25.26	3.02	17.23
	50	1,312.90	22.16	25.52	3.60	22.08	914.60	21.76	23.77	1.80	14.82
	60	929.10	21.94	23.88	2.27	14.70	814.80	21.73	23.35	1.63	12.57
	70	982.00	21.46	22.87	1.39	14.95	834.80	21.75	22.63	0.78	12.34
	10	638.10	56.76	58.07	0.65	102.61	423.10	47.79	50.07	2.75	70.23
	50	1,718.70	47.19	52.38	2.76	277.56	433.00	44.01	44.97	0.96	64.43
	100	2,435.70	44.46	48.76	4.12	386.20	590.70	43.82	44.48	0.99	87.50
`Friedrichshain Center`	200	1,643.60	43.72	45.19	1.39	257.90	712.50	43.65	44.10	0.54	106.37
	300	1,591.00	43.48	44.78	1.21	246.67	624.60	43.59	44.30	0.58	94.07
	400	1,162.60	43.58	44.90	0.66	171.27	442.60	44.10	44.69	0.41	62.35
	500	1,318.80	44.79	45.34	0.40	175.31	667.10	43.88	45.29	1.09	90.23
	10	543.20	79.71	81.94	1.26	210.74	528.30	74.54	81.16	5.46	211.09
	50	1,487.50	70.13	72.08	1.63	571.94	547.70	63.56	64.79	1.16	196.28
	100	2,668.40	63.53	67.09	2.43	1,008.90	640.50	63.09	63.85	0.56	227.44
`Prenzlauerberg Center`	200	2,120.80	62.38	64.92	2.99	764.48	679.10	62.03	63.03	0.71	232.11
	450	1,682.80	62.15	63.67	1.50	572.97	454.50	63.26	64.67	1.11	151.94
	600	1,431.70	63.03	64.44	1.05	472.61	684.00	62.70	64.58	1.23	223.23
	700	2,102.10	64.23	65.26	0.72	653.36	1,136.00	62.53	64.80	1.42	359.48
	10	690.40	62.11	62.79	0.89	194.28	536.60	53.40	53.66	0.21	149.88
	50	1,457.40	55.94	56.98	0.59	412.46	452.80	53.18	53.52	0.19	117.05
	100	2,260.40	54.89	56.05	0.99	650.26	526.10	53.13	53.45	0.28	138.26
`Tiergarten Center`	200	1,946.10	53.86	55.10	0.88	538.28	559.70	53.08	53.36	0.18	140.62
	450	1,529.30	53.20	54.11	0.55	397.31	590.30	53.02	53.21	0.21	147.97
	600	1,388.00	53.46	53.99	0.35	340.03	590.80	53.06	53.36	0.29	139.86
	700	1,620.00	53.84	54.49	0.70	375.46	512.90	53.13	53.45	0.28	119.30

TABLE 7. Detailed results of SPT and SPTF for the BRKGA+LS algorithm.

		SPT					**SPTF**				
Instance	$\mathcal{K}$	**Gens**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**	**Gens**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**
	10	702.60	79.78	82.40	1.68	315.26	356.40	67.94	71.29	2.40	160.00
	50	1,944.60	70.32	70.94	0.53	870.80	613.30	64.42	66.02	1.68	254.33
	100	1,963.10	68.72	70.14	1.22	873.46	659.20	64.41	64.94	0.30	267.42
`Mitte Center`	200	2,251.00	66.41	69.50	3.27	967.11	609.10	64.23	64.97	0.42	245.50
	400	2,300.80	64.33	65.82	1.03	998.54	906.10	64.22	64.56	0.30	365.16
	600	1,770.60	64.00	65.49	0.86	743.23	1,172.80	64.05	64.55	0.28	446.78
	800	2,461.70	65.17	65.69	0.45	937.58	502.20	64.66	65.31	0.51	187.15
	10	427.70	15.39	15.65	0.27	210.72	474.70	12.77	12.87	0.08	241.91
	50	2,260.40	13.96	14.17	0.13	1,110.93	899.50	12.65	12.73	0.10	439.91
	100	3,146.20	13.33	13.68	0.22	1,524.61	644.30	12.73	12.82	0.09	314.96
`Anaheim`	300	3,853.60	14.16	15.15	0.89	1,930.75	1,792.00	13.14	13.75	0.62	895.10
	500	3,664.30	12.82	13.20	0.41	1,849.16	1,411.80	12.96	13.47	0.31	691.70
	700	3,384.50	12.76	13.00	0.23	1,647.53	1,304.30	12.92	13.11	0.10	625.22
	800	2,942.40	12.73	12.94	0.21	1,310.76	1,551.60	12.88	13.15	0.25	684.02
	10	559.60	92.13	92.83	0.54	1,791.12	346.50	73.13	74.72	1.28	1,112.57
	100	1,130.10	79.40	81.50	1.53	3,604.58	658.20	67.42	68.15	0.63	1,984.55
	250	1,112.60	76.14	80.30	3.15	3,604.64	570.40	67.14	67.90	0.47	1,697.06
`MPF Center`	500	1,098.00	82.44	84.71	1.76	3,604.45	722.10	66.90	68.26	1.00	2,143.66
	1000	1,072.60	75.37	77.58	1.51	3,604.69	978.90	67.56	68.58	0.75	2,873.94
	1500	1,096.30	73.76	74.84	0.96	3,604.20	1,042.10	67.89	68.86	0.73	2,992.60
	2000	1,222.00	73.46	74.74	1.07	3,603.84	1,322.60	68.67	69.50	0.62	3,567.67

Table 8. Detailed results of SPT and SPTF for the BRKGA+LS algorithm.

		SPT					**SPTF**				
Instance	$\mathcal{K}$	**Gens**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**	**Gens**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**
	10	518.80	15.84	20.25	5.39	1.898.79	445.00	7.87	8.21	0.34	1.689.84
	100	1.013.50	9.35	10.34	1.44	3.604.46	811.40	7.41	7.47	0.03	3.001.65
	500	989.40	9.54	9.89	0.32	3.607.35	826.10	7.68	8.23	0.49	3.057.13
`Barcelona`	1,000	901.40	13.92	25.44	12.31	3.607.07	921.20	10.67	11.92	1.66	3,523.70
	1500	898.60	9.37	12.85	3.09	3,603.47	974.00	9.75	10.79	0.84	3,604.00
	2,000	923.10	8.53	9.37	0.60	3,602.80	995.50	9.04	9.51	0.37	3,466.61
	2,500	1,092.20	8.08	8.73	0.56	3,603.30	1,091.80	8.33	8.97	0.44	3,422.55
	10	531.60	35.03	41.79	7.07	2,568.40	365.20	17.83	18.20	0.31	1,828.35
	100	735.60	20.36	21.66	1.14	3,608.11	669.80	15.77	16.21	0.23	3,313.55
	500	709.70	21.68	26.47	4.07	3,606.27	725.20	16.66	18.93	1.52	3,602.52
`Winnipeg`	1000	670.80	24.25	48.01	24.35	3,604.98	703.50	23.01	31.43	7.04	3,606.65
	1500	666.70	19.37	30.44	12.14	3,604.68	731.00	20.31	23.71	3.23	3,604.20
	2000	686.10	18.25	20.68	2.38	3,605.43	770.00	18.39	20.50	2.31	3,604.12
	2800	836.50	16.90	18.29	1.40	3,603.79	863.50	17.59	18.30	0.39	3,604.91
	10	593.90	99.43	199.67	130.35	7,214.70	571.70	19.46	20.41	0.95	7,210.80
	100	591.60	22.50	32.00	11.66	7,211.24	574.20	16.05	16.81	0.71	7,210.21
	500	566.60	24.05	27.17	2.64	7,209.27	580.90	17.38	17.80	0.37	7,209.44
`ChicagoSketch`	1000	537.20	42.34	55.09	10.78	7,211.89	581.10	19.50	20.60	1.07	7,208.20
	1500	505.60	105.64	362.14	280.35	7,206.20	576.60	19.43	24.46	5.04	7,206.38
	2000	506.80	29.26	172.01	167.07	7,207.66	589.90	18.42	23.49	5.08	7,208.43
	2900	605.70	17.21	28.21	11.52	7,207.06	656.70	17.24	18.54	1.19	7,208.06

We next explore the main characteristics of the solutions returned by BRKGA+LS. For the best solution found among the five runs, we present in Table 11 the average number of paths for each OD pair (#Path), the average number of hops among all OD shortest paths (#Hop), the average number of tolled arcs among the shortest paths (#Toll), the average sum of the tariffs among the shortest paths (Tariff), and the number of different arcs used in each OD pair (#Arcs).

The column #Path presents the average number of shortest paths used by each OD pair. When we increase $\mathcal{K}$, we observe a reduction in the number of shortest paths, especially with a small $\mathcal{K}$. In this case, considering SPT, there are alternative shortest paths without tolls, and thus only the hop count is used to calculate the shortest path, increasing the possibility of paths with the same cost. On the other hand, for SPTF, adding the free flow time to the arc weight avoids this occurrence and we observe a reduction in the number of shortest paths.

In computer networking, hop count refers to intermediate devices through which a piece of data must pass between source and destination. We extend this concept to road networks where the intermediate devices are vertices which represent intersecting roads or waypoints. Thus, we show in column #Hops the average hop count for all shortest paths between OD pairs. When compared with the situation without tolls ($\mathcal{K} = 0$) we observe that as the number of installed tolls increases, the number of hop counts decreases in SPTF. For the SPT, this number starts increasing, but then decreases after some value of $\mathcal{K}$. For large $\mathcal{K}$, #Hops values are similar for both for both SPT and SPTF.

The column #Tolls shows the average number of tolls installed on OD shortest paths. Clearly, this value increases with $\mathcal{K}$. In the column Tariff we show the average value of tariff in each path. In this problem our objective is to install $\mathcal{K}$ tolls such that congestion costs are minimized. Since minimizing the tariff is not part of the objective function, that results in more variability in this column. In SPT, the impact of a unit of tariff is higher than in SPTF. As can be seen in the last column, we conclude that an increase in $\mathcal{K}$ influences the average number of arcs of the shortest path of OD pairs slightly.

5. Conclusions

In this paper we presented an extensive study of the tollbooth problem. Two mathematical formulations for different versions of the tollbooth problem were presented, as well as linearizations that give lower and upper bounds for their objective functions. Computational tests were conducted taking into account the original and the linearized models, applied on two sets of synthetic and real-world instances. Moreover, a previously proposed biased random-key genetic algorithm was run for this same set of instances. A set of experiments were performed for the sake of results comparison.

When analyzing the results for the mathematical models, we concluded that the model MM2, which includes shortest paths and even split constraints, has a large number of variables and constraints, making it difficult to be solved with general-purpose solvers, even when we limit ourselves to small instances. On the other hand, if shortest paths and even split constraints are removed from the model, giving rise to a simplified version of the problem, the linearized versions of the problem can be solved efficiently by CPLEX. However, results obtained with the

TABLE 9. Detailed results of best solution found by BRKGA+LS algorithm.

		SPT					**SPTF**				
Instance	$\mathcal{K}$	**#Path**	**#Hop**	**#Toll**	**Tariff**	**UArcs**	**#Path**	**#Hop**	**#Toll**	**Tariff**	**UArcs**
	0	1.97	2.51	-	-	4.97	1.05	2.14	-	-	3.24
	10	1.71	2.68	0.20	1.16	4.79	1.07	2.10	0.38	3.54	3.25
	20	1.43	2.54	0.61	1.06	4.33	1.11	2.26	0.71	7.04	3.41
`SiouxFalls`	30	1.24	2.53	1.19	8.13	3.87	1.10	2.21	1.17	8.16	3.32
	40	1.15	2.50	1.42	11.23	3.75	1.16	2.20	1.55	11.53	3.50
	50	1.14	2.27	1.84	12.39	3.50	1.08	2.22	1.92	11.81	3.33
	60	1.11	2.22	2.42	12.26	3.38	1.05	2.18	2.25	13.02	3.28
	70	1.05	2.21	2.79	11.78	3.31	1.09	2.21	2.85	10.13	3.31
	0	1.96	9.36	-	-	11.94	1.52	12.24	-	-	12.82
	10	2.09	10.42	0.10	1.00	12.89	1.59	12.99	0.53	11.95	12.95
	50	1.37	10.20	0.35	5.55	11.91	1.26	11.31	1.48	11.31	12.22
`Friedrichshain Center`	100	1.41	11.93	1.62	10.58	12.42	1.16	11.60	2.15	10.76	12.21
	200	1.18	11.23	3.33	10.99	12.08	1.08	10.86	3.64	11.06	11.96
	300	1.12	11.38	5.70	9.50	12.11	1.00	10.32	5.52	10.29	11.32
	400	1.11	10.88	6.54	9.81	11.98	1.00	10.53	7.32	10.18	11.53
	500	1.09	10.01	10.16	8.86	11.03	1.00	9.95	10.17	9.30	10.95
	0	2.75	14.72	-	-	17.11	1.79	18.67	-	-	18.00
	10	2.63	16.27	0.09	3.00	18.27	1.48	16.55	0.49	11.76	17.57
	50	1.84	15.59	0.39	1.44	17.51	1.46	16.27	1.27	7.77	17.14
`Prenzlauerberg Center`	100	1.73	17.16	1.67	6.39	17.92	1.22	15.45	2.23	10.76	16.76
	200	1.53	16.92	3.40	10.95	17.37	1.16	15.88	4.27	10.05	16.75
	450	1.03	14.82	7.59	10.07	15.91	1.00	15.75	8.37	10.54	16.76
	600	1.01	14.66	10.91	9.39	15.73	1.01	14.48	11.63	10.50	15.54
	700	1.00	14.41	14.40	8.73	15.42	1.00	14.00	14.41	9.94	15.00
	0	1.92	15.40	-	-	17.32	1.12	17.43	-	-	18.51
	10	1.86	16.35	0.00	0.00	18.32	1.07	15.73	0.30	7.72	17.18
	50	1.43	16.83	0.20	3.68	18.46	1.06	15.98	1.03	8.07	17.40
`Tiergarten Center`	100	1.33	17.10	0.52	2.14	19.08	1.06	15.70	2.45	8.52	17.15
	200	1.13	15.92	3.95	9.76	17.71	1.03	16.08	4.12	9.61	17.32
	450	1.00	15.90	8.02	10.81	16.94	1.01	15.90	8.77	11.68	16.91
	600	1.00	15.56	11.30	10.28	16.59	1.00	15.89	11.75	10.60	16.89
	700	1.00	15.63	13.46	8.86	16.63	1.00	15.57	14.60	10.08	16.57

TABLE 10. Detailed results of best solution found by BRKGA+LS algorithm.

		SPT					SPTF				
Instance	$\mathcal{K}$	**#Path**	**#Hop**	**#Toll**	**Tariff**	**UArcs**	**#Path**	**#Hop**	**#Toll**	**Tariff**	**UArcs**
	0	2.79	14.95	-	-	17.75	1.33	17.49	-	-	17.81
	10	3.10	17.00	0.00	0.00	18.91	1.41	17.18	0.58	9.47	17.47
	50	1.54	15.74	0.15	3.45	17.39	1.24	16.02	1.66	11.45	16.64
`Mitte Center`	100	1.50	15.51	0.65	4.88	17.71	1.29	16.66	2.78	10.40	17.28
	200	1.34	16.93	3.20	9.51	17.62	1.20	16.65	3.61	10.22	17.00
	400	1.06	15.88	6.75	10.62	16.75	1.10	15.97	6.57	11.88	16.95
	600	1.03	15.47	9.71	9.35	16.50	1.00	15.25	10.37	10.82	16.25
	800	1.02	14.73	13.29	9.71	15.75	1.04	14.85	13.90	10.58	15.87
	0	8.71	15.64	-	-	21.45	1.35	15.67	-	-	17.46
	10	16.79	19.71	0.00	0.00	23.19	1.40	15.43	0.13	4.28	17.09
	50	10.12	19.34	0.00	0.00	22.90	1.42	15.31	0.55	7.82	16.89
`Anaheim`	100	5.17	19.19	0.05	9.00	21.58	1.55	16.31	0.91	10.50	17.31
	300	3.19	18.07	3.03	8.59	19.07	1.68	15.45	3.23	8.73	16.67
	500	1.02	14.80	6.45	9.15	15.91	1.00	15.12	6.90	9.50	16.13
	700	1.03	14.46	9.64	9.88	15.56	1.09	14.29	10.25	11.84	15.23
	800	1.04	14.05	11.97	9.92	15.11	1.04	13.84	12.25	10.94	14.88
	0	5.09	24.35	-	-	27.54	2.28	31.51	-	-	29.17
	10	5.05	25.54	0.00	0.00	27.62	2.31	31.89	0.25	7.25	28.94
	100	2.45	23.78	0.00	0.00	26.93	1.72	28.52	1.55	8.49	27.84
`MPF Center`	250	2.73	25.36	0.06	4.95	27.50	1.47	27.81	2.88	9.31	27.39
	500	2.68	31.56	4.49	7.70	28.54	1.34	27.49	4.77	10.30	27.24
	1000	1.29	27.32	8.47	9.45	26.81	1.11	25.30	8.86	10.42	25.98
	1500	1.02	23.70	13.78	9.82	24.62	1.00	23.98	15.14	10.37	24.98
	2000	1.01	22.12	19.35	9.53	23.16	1.03	23.49	20.62	10.20	24.41

Table 11. Detailed results of best solution found by BRKGA+LS algorithm.

		SPT					SPTF				
Instance	$\mathcal{K}$	**#Path**	**#Hop**	**#Toll**	**Tariff**	**UArcs**	**#Path**	**#Hop**	**#Toll**	**Tariff**	**UArcs**
	0	7.37	15.85	-	-	20.82	1.16	18.73	-	-	20.73
	10	6.90	18.21	0.00	1.00	24.76	1.19	19.40	0.07	1.00	21.31
	100	5.31	18.60	0.07	2.89	24.34	1.15	19.73	0.29	3.95	21.45
`Barcelona`	500	6.32	22.37	0.16	7.71	26.17	1.09	20.50	0.85	2.97	22.07
	1000	3.68	29.60	4.60	4.29	27.15	1.20	19.96	3.59	4.65	21.12
	1500	1.13	19.54	8.66	7.59	20.30	1.11	19.35	8.61	7.19	19.91
	2000	1.06	16.81	11.74	8.79	18.03	1.01	16.66	12.07	8.58	17.63
	2500	1.13	16.09	16.68	8.80	16.95	1.00	14.93	15.57	8.96	15.94
	0	3.92	20.72	-	-	25.96	1.04	24.98	-	-	25.73
	10	3.90	21.70	0.00	0.00	26.77	1.06	25.16	0.19	1.22	25.54
	100	4.58	26.78	0.05	7.39	30.25	1.08	24.97	0.91	1.94	25.37
`Winnipeg`	500	2.96	28.79	0.46	5.99	30.41	1.11	26.09	2.81	3.95	26.55
	1000	1.67	26.98	5.88	7.11	26.82	1.03	25.81	6.18	6.56	26.81
	1500	1.18	24.39	9.75	9.04	25.13	1.05	23.97	9.24	8.26	24.83
	2000	1.10	22.21	13.02	9.37	22.98	1.04	21.84	13.78	9.04	22.60
	2800	1.07	19.98	20.41	9.56	21.02	1.02	19.71	20.19	9.93	20.64
	0	20.04	14.86	-	-	23.69	1.01	12.30	-	-	13.32
	10	21.17	15.24	0.00	0.00	24.34	1.01	12.20	0.09	2.60	13.22
	100	19.89	15.59	0.04	9.94	24.75	1.01	12.12	0.54	2.77	13.14
`ChicagoSketch`	500	11.22	15.67	0.42	11.39	22.64	1.01	12.33	1.67	4.11	13.33
	1000	4.12	15.94	0.91	9.54	19.87	1.00	12.56	3.13	6.31	13.58
	1500	1.84	18.19	4.87	5.43	17.72	1.02	12.68	5.42	7.12	13.63
	2000	1.34	14.62	8.19	7.88	15.06	1.00	12.28	7.58	8.49	13.28
	2900	1.15	12.10	12.77	9.00	13.13	1.00	11.69	12.34	8.60	12.69

biased random-key genetic algorithm for the complete model point to it as having the best tradeoff between computation time and solution quality on this problem.

Finally, considering that users naturally take the least costly path, toll setting can be used to better distribute the flow in the network and consequently reduce traffic congestion.

Acknowledgment

This work has been partially supported by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) and FAPERGS (Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul).

References

L. Bai, D.W. Hearn, and S. Lawphongpanich. Decomposition techniques for the minimum toll revenue problem. *Networks*, 44(2):142–150, 2004. doi: 10.1002/net.20024.

L. Bai, D.W. Hearn, and S. Lawphongpanich. A heuristic method for the minimum toll booth problem. *Journal of Global Optimization*, 48:533–548, 2010. ISSN 0925-5001. URL `http://dx.doi.org/10.1007/s10898-010-9527-7`. 10.1007/s10898-010-9527-7.

H. Bar-Gera. Transportation networks test problems, 2012. URL `http://www.bgu.ac.il/~bargera/tntp`.

J.C. Bean. Genetic algorithms and random keys for sequencing and optimization. *ORSA J. on Comp.*, 6:154–160, 1994.

M.J. Beckmann, C.B. McGuire, and C.B. Winsten. *Studies in the economics of transportation*. Yale University Press, 1956.

P. Broström and K. Holmberg. Multiobjective design of survivable ip networks. *Annals of Operations Research*, 147:235–253, 2006. ISSN 0254-5330. URL `http://dx.doi.org/10.1007/s10479-006-0067-y`. 10.1007/s10479-006-0067-y.

Bureau of Public Roads. Bureau of public roads: Traffic assignment manual. *US Department of Commerce, Urban Planning Division*, 1964.

L.S. Buriol, M.G.C. Resende, C.C. Ribiero, and M. Thorup. A hybrid genetic algorithm for the weight setting problem in OSPF/IS-IS routing. *Networks*, 46: 36–56, 2005.

L.S. Buriol, M.H. Hirsch, P.M. Pardalos, T. Querido, M.G.C. Resende, and M. Ritt. A biased random-key genetic algorithm for road congestion minimization. *Optimization Letters*, 4:619–633, 2010. ISSN 1862-4472. URL `http://dx.doi.org/10.1007/s11590-010-0226-6`. 10.1007/s11590-010-0226-6.

R.B. Dial. Minimal-revenue congestion pricing part II: An efficient algorithm for the general case. *Transportation Research Part B*, 34:645–665, 1999a.

R.B. Dial. Minimal-revenue congestion pricing part I: A fast algorithm for the single origin case. *Transportation Research Part B*, 33:189–202, 1999b.

M. Ericsson, M. G. C. Resende, and P. M. Pardalos. A genetic algorithm for the weight setting problem in OSPF routing. *Journal of Combinatorial Optimization*, 6:299–2002, 2002.

B. Fortz and M. Thorup. Increasing internet capacity using local search. *Computational Optimization and Applications*, 29(1):189–202, 2004.

J.F. Gonçalves and M.G.C. Resende. Biased random-key genetic algorithms for combinatorial optimization. *J. of Heuristics*, 17:487–525, 2011.

J.F. Gonçalves, M.G.C. Resende, and R.F. TOso. Biased and unbiased random-key genetic algorithms: An experimental analysis. Technical report, AT&T Labs Research, Florham Park, New Jersey, December 2012.

D.W. Hearn and M.V. Ramana. Solving congestion toll pricing models. *Equilibrium and Advanced Transportation Modeling*, pages 109–124, 1998. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.4999`.

D.W. Hearn and M.B. Yildrim. A toll pricing framework for pricing assignment problems and elastic demands. In M. Gendreau and P. Marcotte, editors, *Transportation and network analysis: current trends*, volume 63 of *Applied Optimization.* Springer, 2002. URL `http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.141.2739`.

D. Schrank, T. Lomax, and B. Eisele. 2011 urban mobility report. Technical report, Texas Transportation Institute, 2011. URL `http://mobility.tamu.edu/files/2011/09/congestion-cost.pdf`.

W. M. Spears and K. A. DeJong. On the virtues of parameterized uniform crossover. In *Proceedings of the Fourth International Conference on Genetic Algorithms*, pages 230–236, 1991.

T. Tsekeris and S. Voß. Design and evaluation of road pricing: state-of-the-art and methodological advances. *Netnomics*, 10:5–52, 2009. ISSN 1385-9587. URL `http://dx.doi.org/10.1007/s11066-008-9024-z`. 10.1007/s11066-008-9024-z.

W. Wen. A dynamic and automatic traffic light control expert system for solving the road congestion problem. *Expert Systems with Applications*, 34(4):2370–2381, 2008. ISSN 0957-4174. doi: 10.1016/j.eswa.2007.03.007. URL `http://www.sciencedirect.com/science/article/pii/S0957417407001303`.

H. Yang and X. Zhang. Optimal toll design in second-best link-based congestion pricing. *Transportation Research Record: Journal of the Transportation Research Board*, 1857(1):85–92, 2003. doi: 10.3141/1857-10.

(Fernando Stefanello) Instituto de Informática, Universidade Federal do Rio Grande do Sul, 91501-970 Porto Alegre, RS, Brazil
E-mail address: `fstefanello@inf.ufrgs.br`

(Luciana S. Buriol) Instituto de Informática, Universidade Federal do Rio Grande do Sul, 91501-970 Porto Alegre, RS, Brazil
E-mail address: `buriol@inf.ufrgs.br`

(Michael J. Hirsch) Raytheon Company Intelligence and Information Systems, 300 Sentinel Drive, Annapolis Junction, MD 20701, USA.
E-mail address, Michael J. Hirsch: `mjh8787@ufl.edu`

(Panos M. Pardalos) Department of Industrial and Systems Engineering, University of Florida, 303 Weil Hall, Gainesville, FL 32611, USA.
E-mail address: `pardalos@ufl.edu`

(Tania Querido) Linear Options Consulting, 7450 SW 86th Way, Gainesville, FL 32608, USA.
E-mail address: `tania@linearoptions.com`

(Mauricio G.C. Resende) Algorithms and Optimization Research Department, AT&T Labs Research, 180 Park Avenue, Room C241, Florham Park, NJ 07932 USA.
E-mail address: `mgcr@research.att.com`

(Marcus Ritt) Instituto de Informática, Universidade Federal do Rio Grande do Sul, 91501-970 Porto Alegre, RS, Brazil
E-mail address: `mrpritt@inf.ufrgs.br`