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 biased random-key genetic algorithm on a set of real-world instances, analyzing solutions when computing shortest paths according to two different weight functions. 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: March 31, 2014. Revised July 7, 2014, October 21, 2014.

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). In fact, tolls are being deployed for traffic engineering in many small as well as large cities around the world.

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. 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 is 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 Yildirim, 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).

Two important transportation network concepts were introduced by Wardrop (1952): user equilibrium (UE) and system optimal (SO). The former is related to the equilibrium obtained when each user chooses a route that minimizes his/her costs in a congested network. In an UE state, any user can reduce his/her own travel cost by changing routes. Differently, SO is related to a state of equilibrium with minimum average journey time. This occurs when the users cooperate to choose their routes. However, the user usually chooses his/her own route in a non-cooperative manner. In a simplistic modeling behavior, users can choose their routes by different criteria. One possible simplification assumes that users choose their routes considering only fixed costs such as time to travel, or a value that depends on the congestion, or even only the toll values. These situations do not correspond to user equilibrium, but model different behaviors of the users.

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 calculate shortest paths according to two weight functions. In the first, the weights correspond to the tariffs of the tolled arcs. The second function considers as the weight of each arc its toll tariff added to its free flow time, where

[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.

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, considering a new arc value to calculate shortest paths, a review of the algorithm components, such as the local search, and a more detailed review of the behavior of the algorithm, including a new set of instances and an analysis of characteristics of the final solutions.

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 random-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 α_a and β_a, two real-valued arc-tuning parameters. In addition, let

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

denote the set of commodities or origin-destination (OD) pairs, where $o(k)$ and $d(k)$ represent, respectively, the origination and destination nodes for $k = 1, \ldots, |K|$. Each commodity 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 levying 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 of the traffic congestion cost can be defined in different ways according to specific goals. In this paper we use the potential function

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

which is the convex travel cost function of the Bureau of Public Roads (1964) for measuring link congestion scaled by the term $\ell_a/\mathcal{S}$. This way, the potential function evaluates the average user travel time over all trips. 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

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

subject to:

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

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

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

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

Its goal is to determine flows on each arc such that the average user travel time 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 $a \in A$. 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.

This model computes flow distribution without taking into account that users take a least cost route, providing a lower bound for the tollbooth problem to be described in the next subsection.

2.2. **Model for the tollbooth problem (MM2).** A mathematical programming model for the tollbooth problem is

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

subject to:

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

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

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

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

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

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

$$M_3 y_a^q \geq x_a^q,\ \forall a\in A,\ \forall q\in\mathcal{Q}, \tag{13}$$

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

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

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

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

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

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

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

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

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, or the sum of tolls and free flow times. We later describe these arc weight functions in more detail.

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 $w_a \in \{0, P_l, P_l+1, \ldots, P_u\}$ the toll tariff levied on arc $a\in A$, where $P_l, P_u \in \mathbb{N}^+$ are the minimum and maximum tariff values, respectively. For convenience we define $P_l = 1$. If no toll is levied on arc a, then $w_a = 0$. The binary decision variable $p_a = 1$ if a tollbooth is deployed on arc $a\in A$. The auxiliary binary variable $y_a^q = 1$ 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 $q\in\mathcal{Q}$, and the constants M_1, M_2, and M_3 are sufficiently larger numbers.

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. The other constraints force the flow of each commodity to follow the shortest path between the corresponding OD pair. An arc a belongs to the shortest path to destination q if the distance $\delta_{a_h}^q - \delta_{a_t}^q$ is equal to the arc cost, which in this case is $C_a + w_a$, where C_a will be introduced later in this subsection. Thus, 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 have reduced cost $C_a + w_a + \delta_{a_h}^q - \delta_{a_t}^q > 0$. 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. $y_a^q = 1$. In the computational experiments of Subsection 4.2, we used $M_1 = 100$ and $M_2 = 1000$. Constraints (13) assure that flow is sent only on arcs belonging to a shortest path. Constraints (14) are the even-split constraints. They 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. if $y_a^q = y_b^q = 1$, then the flow on these arcs must be evenly split, i.e. $x_a^q = x_b^q$. To see this, suppose $y_a^q = y_b^q = 1$. The constraint for pair $\{a, b\} \in \mathsf{A}^2_{OUT(v)}$ implies that $x_a^q \leq x_b^q$. By symmetry the constraint for pair $\{b, a\} \in \mathsf{A}^2_{OUT(v)}$ implies that $x_a^q \geq x_b^q$. Consequently, $x_a^q = x_b^q$. Note that taking $M_3 = \max_{q \in \mathcal{Q}} \left(\sum_{v \in V} d_{v,q} \right)$ 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_a^k or y_b^k equal to one.

A model for OSPF routing, which also considers 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 according to two weight functions. 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 $C_a = \epsilon$, a 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 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, resulting in an approximation of the nonlinear objective function. One possible option is to

approximate the nonlinear function by a piecewise linear function. Fortz and Thorup (2004) proposed a piecewise-linear function for a general routing problem to approximate network congestion cost. Ekström et al. (2012) describe an iterative approximation by piecewise linear function for the travel time and total travel time, resulting in a mixed integer linear program.

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 nonlinear 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. 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

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

subject to:

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

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

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

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

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

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 $X_0, X_1, \ldots, X_n$ 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 save time while solving the model since each element entails $|A|$ additional constraints.

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 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 $i = 1, \ldots, n$, as

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

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

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 $X_0, X_1, \ldots, X_n$ be the values for which the approximation is computed. If $\ell_a / c_a \leq X_n, \forall a \in A$, then $\varphi^l \leq \Phi \leq \varphi^u$.*

Proof. As Φ is convex, 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 Proposition 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 proposition 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 piecewise-linear cost functions φ^u and φ^l for an arc $a \in A$ with $t_a = 5$, $c_a = 200$, $\alpha_a = 4$, $\beta_a = 0.15$, 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 dense concentration of points in this region is used because the flow on the majority of the arcs is concentrated around their capacity. Thus, to obtain a good approximation requires that several X values be set to values around $\frac{l_a}{c_a} = 1$. Note that a ratio of $\frac{l_a}{c_a} > 1$ indicates that the arc is overloaded.

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 for 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 a deterministic algorithm

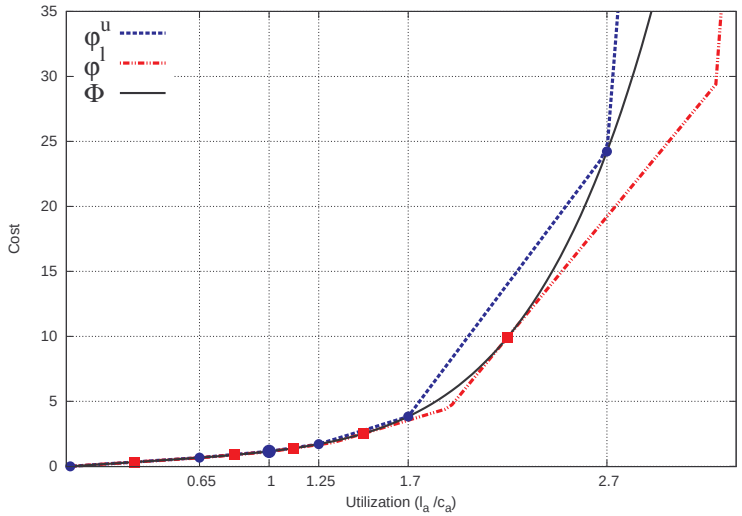

FIGURE 1. Comparison of the cost function with the linear piecewise-linear cost function.

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 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 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. (2014) 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 the encoding and decoding procedures next and give values for parameters as well as the stopping criterion in Section 4.

Solutions are encoded as a $2 \times |A|$ vector $\mathcal{X}$, where $|A|$ is the cardinality of the set A of arcs in the network. The first $|A|$ keys correspond to the random keys which define the toll tariffs while the last $|A|$ keys correspond to a binary vector b, with $\mathcal{K}$ positions set to one, used to indicated tolled arcs.

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 $b_a \cdot \lceil \mathcal{X}_a \cdot w_{\max} \rceil$. In an initial solution, the $\mathcal{K}$ tolled arcs are selected randomly by uniform distribution. In a crossover, if both parents have a toll in arc a, the same arc is tolled in the child. The remaining tolls are selected randomly among the arcs whose parents have different values.

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 for 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 a local improvement. Local search is applied to the solution produced in the first phase of the decoder. In short, it works as follows. Let q_{ls} be an integer parameter and $A^* \subseteq A$ be the $q = \min\{|A|, q_{ls}\}$ 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^*.

In the local improvement, every time a weight is changed (added by one unit, inserted or removed) the current shortest paths are updated (Buriol et al., 2008) instead of recomputed from scratch, thus saving a considerable amount of running time.

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 MM2 with piecewise linear function, which considers the shortest-path 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 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 ten real-world instances (*S2*) considered in our experiments and made available by Bar-Gera (2013).

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

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

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_h 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`

TABLE 1. Attributes for the instances are given in each column. For each instance, its row lists the set identification (*S1* or *S2*), instance name, number of vertices, links, OD 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

In the first two columns after the name of the instance, we present the objective function values Φ and the computational times for model MM1 obtained with the nonlinear solver MOSEK 6.0 using the modeling system GAMS[4]. A few nonlinear solvers are part of the GAMS system and we evaluated the performance of all of them. Some of them are general nonlinear 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 times 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 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 φ^l and φ^u have values that are, respectively, close to $\Phi\{\varphi^l\}$ and $\Phi\{\varphi^u\}$. 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 times spent by MOSEK on the convex function are two to four 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 (computing 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

[4] `www.gams.com`

TABLE 2. Computational results for MM1 and LMM1.

	MM1		LMM1-lower estimate			LMM1-upper estimate		
Instance	Φ	**Time(s)**	φ^{l}	$\Phi\{\varphi^{\mathrm{l}}\}$	**Time(s)**	φ^{u}	$\Phi\{\varphi^{\mathrm{u}}\}$	**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

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-split constraints (Eqs. 14) 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 for the full model (MM2).

4.2. **Results for the tollbooth problem with piecewise-linear cost (LMM2).** This set of experiments tests the performance of CPLEX on MM2, the model that includes shortest paths and even-split constraints. We run the model considering both weight functions to calculate shortest paths (SPT and SPTF) and both piecewise-linear functions introduced in Section 2.3.

Table 3 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 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.

Table 3 illustrates 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 not even a feasible solution was found in this time limit. A small increase in the instance size implies in a large increase in the computational effort spent to solve the model. 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 SPTF. 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 a less than one second, 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 shortest-path evaluations SPT and SPTF, since the solver was unable to find any feasible solution within the time limit.

In summary, Table 2 shows that solving the simplified model MM1, i.e. MM2 without the shortest paths computation and even-load constraints, takes a considerable time, while their corresponding linearized versions φ^l and φ^u are calculated very quickly for almost all cases. Table 3, on the other hand, shows 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 a heuristic solution to solve the tollbooth problem, and the results of the proposed biased random-key genetic algorithm are presented in the next subsection.

TABLE 3. Computational results for LMM2 to SPT and SPTF.

			Approx.		Obj. Function		Solver *gap*		Time(s)	
Type	**Instance**	$\mathcal{K}$	$\varphi^{\mathbf{l}}$	$\varphi^{\mathbf{u}}$	$\mathbf{\Phi}\{\varphi^{\mathbf{l}}\}$	$\mathbf{\Phi}\{\varphi^{\mathbf{l}}\}$	$\varphi^{\mathbf{l}}$	$\varphi^{\mathbf{u}}$	$\varphi^{\mathbf{l}}$	$\varphi^{\mathbf{u}}$
SPT	SiouxFalls_09	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
		10	6.27	6.47	6.33	6.33	0.00	0.00	4.1	35.2
		15	6.27	6.46	6.32	6.32	0.00	0.00	1.3	15.3
		20	6.27	6.46	6.32	6.32	0.00	0.00	0.4	3.5
	SiouxFalls_10	3	-	-	-	-	-	-	1,800.0	1,800.0
		7	-	-	-	-	-	-	1,800.0	1,800.0
		14	-	-	-	-	-	-	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
		28	6.70	6.90	6.76	6.78	0.23	0.21	1,800.0	1,800.0
	SiouxFalls_12	3	-	-	-	-	-	-	1,800.0	1,800.0
		7	-	-	-	-	-	-	1,800.0	1,800.0
		15	-	-	-	-	-	-	1,800.0	1,800.0
		22	-	-	-	-	-	-	1,800.0	1,800.0
		30	-	-	-	-	-	-	1,800.0	1,800.0
	SiouxFalls_14	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
		14	-	-	-	-	-	-	1,800.0	1,800.0
		21	46.14	-	64.96	-	0.49	-	1,800.0	1,800.0
		28	46.14	-	64.96	-	0.39	-	1,800.0	1,800.0
SPTF	SiouxFalls_09	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
		10	6.27	6.46	6.32	6.32	0.00	0.00	0.2	573.3
		15	6.27	6.46	6.32	6.32	0.00	0.00	0.4	0.4
		20	6.27	6.46	6.32	6.32	0.00	0.00	0.4	0.4
	SiouxFalls_10	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
		14	-	6.90	-	6.78	-	0.31	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
		28	6.70	6.90	6.76	6.78	0.04	0.51	1,800.0	1,800.0
	SiouxFalls_12	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
		15	-	-	-	-	-	-	1,800.0	1,800.0
		22	-	-	-	-	-	-	1,800.0	1,800.0
		30	11.18	-	11.66	-	1.64	-	1,800.0	1,800.0
	SiouxFalls_14	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
		14	-	-	-	-	-	-	1,800.0	1,800.0
		21	-	-	-	-	-	-	1,800.0	1,800.0
		28	-	119.65	-	64.99	-	0.13	1,800.0	1,800.0

4.3. **Results for the biased random-key genetic algorithm.** This section presents results for the biased random-key genetic algorithm applied on instances from class *S2*. We extended the experimental study performed by Buriol et al. (2010) in which results for only three of these instances were presented. Moreover, we provide an analysis of the best solution for each combination of instance, value of $\mathcal{K}$, and problem type (SPT or SPTF).

To tune the parameters, a set of experiments was performed. The experiment consisted of two steps. In the first step, we determined the fixed running time for each triplet: instance (`SiouxFalls`, `Prenzlauerberg Center`, and `Anaheim`), value of $\mathcal{K}$, and problem type (SPT or SPTF). To define the fixed time, we ran the BRKGA with local search using a set of predefined parameters: population size $p = 50$, elite set of size $p_e = 0.25p$, mutant set of size $p_m = 0.05p$, elite key inheritance probability $\rho_A = 0.7$, and a restart parameter $r = 10$. At every r generations we verify whether the best three individuals in the population have identical fitness (within 10^{-3} of each other). If they do, then the second and third best are replaced by two new randomly-generated solutions. The BRKGA was

run for at least 500 and at most 2000 generations, stopping after 100 generations without improvement of the incumbent solution. The fixed time is defined to be the average of five independent runs.

The running time defined in the first step of the tuning phase is used in the second step to determine the best combination of parameter values. We ran the BRKGA for this fixed amount of time with parameters taken from the sets of values shown in the third column of Table 4. All combinations of parameter values were considered.

TABLE 4. Parameter values in tuning experiment.

Description	**Parameter**	**Values**
Population size	p	{50,100}
Elite size	p_e	{0.15p, 0.25p}
Mutation size	p_m	{0, 0.05p, 0.10p}
Inheritance probability	ρ_A	{0.5, 0.7}
Restart	r	{0, 10}
Local Search	q_{ls}	{0*, 2, 5, 10}

*Indicates that no local search is applied.

Given a set of triples, each consisting of an instance, a value of $\mathcal{K}$, and a problem type (SPT or SPTF), we run the BRKGA on each triple using all combinations of the parameters in Table 4. The relative gaps of the fitness values from each run to the best fitness over all runs for each triple is computed. We observe that using a local search in the BRKGA results in better solutions than using no local search. In the case of $q_{ls} = 0$, the average relative gap is 29.86, while for $q_{ls} = 2, 5,$ and 10, the relative gap was 6.70, 5.96, and 5.88, respectively. Therefore, we analyze the remaining parameters considering only runs where $q_{ls} = 10$. Table 5 shows the average relative gaps for these remaining parameters. The best observed parameter values were $p = 100$ for population size, $p_e = 0.15p$ for elite population size, $p_m = 0.05p$ for mutant population size, $\rho_A = 0.70$ for inheritance probability, and $r = 10$ for restart.

TABLE 5. Average of relative gaps obtained for different parameters.

Description	**Parameter**	**Value**	**Gap**
Population size	p	50	6.04
		100	**5.72**
Elite size	p_e	**0.15p**	**5.82**
		0.25p	5.93
Mutation size	p_m	0	6.39
		0.05p	**5.49**
		0.10p	5.76
Inheritance probability	ρ_A	0.5	5.91
		0.7	**5.85**
Restart	r	0	6.23
		10	**5.53**

Once the parameters were set, we ran the BRKGA with local search (BRKGA+LS) with a time limit of 3600 seconds (except for `ChicagoSketch`, the largest instance,

for which we ran with a time limit of 7200 seconds). We allow the maximum number of generations to be 2000, and the maximum number of generations without improvement to be 100.

Table 6 shows averages over five runs of BRKGA+LS and a comparison between SPT and SPTF. For each value of $\mathcal{K}$, it lists the best solution value (Best Φ) over the five runs, average fitness value (Avg Φ), standard deviation (SD), and average running time in seconds.

TABLE 6. Detailed results of SPT and SPTF for BRKGA+LS.

		SPT				SPTF			
Instance	$\mathcal{K}$	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**	**Best Φ**	**Avg Φ**	**SD**	**Time (s)**
SiouxFalls	10	52.38	54.48	2.88	34.83	25.19	25.19	0.00	27.57
	20	32.14	38.01	6.08	59.52	22.72	22.86	0.10	30.26
	30	27.05	28.37	1.90	68.43	22.10	22.30	0.17	32.34
	50	21.59	21.90	0.27	41.81	21.64	21.83	0.18	48.74
	70	21.38	21.53	0.15	32.39	21.55	21.94	0.23	17.18
Friedrichshain Center	10	56.44	56.80	0.34	263.53	46.38	47.23	0.68	218.38
	50	46.59	48.32	2.44	526.92	43.45	43.52	0.06	226.39
	100	43.52	43.93	0.37	710.33	43.41	43.50	0.07	273.65
	300	42.90	43.45	0.33	395.85	43.38	43.59	0.14	209.36
	500	44.04	44.46	0.38	207.16	43.56	44.01	0.37	236.48
Prenzlauerberg Center	10	79.19	80.18	0.99	924.55	65.99	66.13	0.14	682.22
	50	68.21	69.95	1.80	1,780.96	61.72	62.61	0.75	890.95
	100	63.31	63.72	0.26	1,344.99	61.58	61.65	0.07	785.03
	450	61.82	62.18	0.26	1,015.67	61.63	61.81	0.16	735.60
	700	62.42	63.26	0.81	1,066.42	63.03	63.68	0.44	703.19
Tiergarten Center	10	61.97	62.00	0.07	418.51	53.24	53.28	0.06	401.85
	50	56.45	56.69	0.23	1,148.33	52.88	52.92	0.03	534.28
	100	54.14	54.79	0.47	1,499.94	52.92	53.04	0.13	505.92
	450	53.11	53.26	0.13	764.01	52.87	52.93	0.06	444.03
	700	53.60	53.98	0.25	578.06	52.97	53.10	0.12	331.22
Mitte Center	10	79.78	80.11	0.26	805.12	65.84	66.47	0.38	807.10
	50	69.74	70.58	0.67	1,880.73	63.86	64.03	0.21	1,096.38
	100	68.39	68.71	0.33	2,233.21	63.75	63.94	0.19	1,536.11
	400	63.94	64.11	0.15	2,323.10	63.90	64.17	0.22	955.07
	800	64.19	64.47	0.34	1,102.89	63.96	64.30	0.22	879.13
Anaheim	10	15.39	15.42	0.01	785.38	12.72	12.73	0.01	1,328.61
	50	14.01	14.05	0.04	2,611.93	12.58	12.60	0.02	2,058.35
	100	13.41	13.54	0.09	3,406.09	12.58	12.60	0.01	2,034.47
	500	12.73	12.89	0.11	3,602.11	12.62	12.68	0.05	3,601.32
	800	12.60	12.65	0.05	3,096.01	12.57	12.63	0.05	3,041.08
MPF Center	10	91.64	92.03	0.28	3,616.48	70.73	71.14	0.24	3,615.19
	100	82.28	82.62	0.51	3,616.98	66.64	66.73	0.07	3,611.59
	250	82.54	82.84	0.25	3,616.24	66.76	66.93	0.12	3,615.17
	1000	75.08	75.89	0.90	3,612.88	67.92	68.39	0.38	3,611.87
	2000	71.21	72.58	0.90	3,607.61	68.11	68.58	0.30	3,602.20
Barcelona	10	15.84	15.84	0.00	3,622.16	7.82	7.91	0.13	3,618.36
	100	9.41	9.48	0.05	3,622.80	7.25	7.26	0.01	3,613.41
	500	9.62	9.87	0.23	3,619.05	8.15	8.24	0.13	3,613.71
	1500	9.65	10.32	0.43	3,615.32	9.20	9.40	0.13	3,612.30
	2500	8.05	8.23	0.19	3,605.98	7.85	8.00	0.13	3,607.16
Winnipeg	10	32.34	35.22	2.09	3,627.08	17.45	17.59	0.10	3,625.19
	100	20.41	20.90	0.41	3,627.86	15.50	15.62	0.09	3,619.40
	500	26.76	31.68	4.59	3,629.67	19.45	19.69	0.17	3,617.72
	1500	20.34	21.70	1.02	3,616.06	18.96	19.92	0.87	3,618.39
	2800	16.67	16.72	0.06	3,607.96	16.04	16.49	0.29	3,606.33
ChicagoSketch	10	100.18	100.30	0.07	7,267.29	19.24	19.44	0.12	7,254.08
	100	22.14	22.58	0.41	7,257.85	16.62	16.70	0.05	7,268.57
	500	22.77	24.29	0.96	7,268.40	17.99	18.25	0.26	7,277.62
	1500	76.87	154.37	62.94	7,243.13	19.27	20.46	0.77	7,246.76
	2900	16.95	17.51	0.45	7,218.88	15.72	16.04	0.20	7,212.14

The first observation is that as the value of $\mathcal{K}$ increases, the value of Φ tends to decrease and have less variance. In fact, in most cases, the best solutions were found for $\mathcal{K} \geq \frac{|A|}{2}$. With small $\mathcal{K}$ it is easy for flow to bypass tolled arcs, which impairs

traffic engineering. On the other hand, the search space increases considerably for larger $\mathcal{K}$ values, making the problem hard to solve. Since there are $\binom{|A|}{\mathcal{K}}$ configurations for the location of $\mathcal{K}$ tolls and for each configuration each toll can have 20 different values, then the size of the solution space is $\sigma(\mathcal{K}) = \binom{|A|}{\mathcal{K}}20^{\mathcal{K}}$. Thus, the solution space size is much larger for $\mathcal{K} \geq \frac{|A|}{2}$ than for $\mathcal{K} < \frac{|A|}{2}$. Furthermore, even though the maximum of $\sigma(\mathcal{K})$ is achieved for a value of $\mathcal{K} < |A|$, in all of the instances, $\sigma(\mathcal{K}') > \sigma(\mathcal{K})$ for all $\mathcal{K}' > \mathcal{K}$, where $\mathcal{K}'$ is the largest $\mathcal{K}$ value tested. For example, for the `SiouxFalls` instance, for which the largest value of $\mathcal{K}$ tested was 70, $\sigma(\mathcal{K}) = \binom{76}{\mathcal{K}}20^{\mathcal{K}}$, which achieves a maximum for $\mathcal{K} = 73$.

In most entries of Table 6 the standard deviation is small, showing robustness of the algorithm. The table also shows that for small values of $\mathcal{K}$, SPTF has smaller Φ than SPT. This occurs because, for small values of $\mathcal{K}$, SPT has many zero-cost paths, making it difficult to influence flow distribution with tolls.

TABLE 7. Approximation of the lower bound with tolls.

Instance	**Lower Bound**	**BRKGA+LS**
`SiouxFalls`	19.95	21.38
`Friedrichshain Center`	42.47	42.90
`Prenzlauerberg Center`	59.90	61.57
`Tiergarten Center`	52.57	52.87
`Mitte Center`	62.36	63.59
`Anaheim`	12.46	12.57
`MPH Center`	65.88	66.64
`Barcelona`	6.87	7.25
`Winnipeg`	13.67	15.50
`ChicagoSketch`	14.24	15.72

Table 7 presents, for each instance, the shortest average user travel time using tolls obtained by BRKGA in comparison with the optimal distribution flow obtained by solving linear program MM1. An optimal solution for MM1 is a lower bound for the tollbooth problem. The results show that with tolls it is possible to obtain a near-optimal flow distribution.

We next explore the main characteristics of the near-optimal solutions found by BRKGA+LS. For the best solution found in the five runs, Table 8 lists the average number of paths for each OD pair (#Path), the average number of hops among all OD shortest paths (#Hop), the average sum, over all OD pairs, of the tariffs on the shortest paths (#Toll), and the average number of distinct arcs used, over all OD pairs (#UArc).

Columns #Path in Table 8 show that when $\mathcal{K}$ increases, a strong reduction in the number of shortest paths is observed for SPT, while for SPTF, the reduction is not as pronounced. Again, this occurs because of the large number of zero-cost paths present in SPT when $\mathcal{K}$ is small. Of these, traffic flows on one or more paths of minimum hop count. On the other hand, for SPTF, the inclusion of free flow time to the arc weight leads to paths of distinct cost, with a few of minimum cost (in many cases a single minimum cost path).

Columns #Hop in Table 8 show the minimum hop count distance between OD vertices. For large values of $\mathcal{K}$ we observe that as the number of installed tolls increases, the hop count decreases in both SPT and SPTF. This occurs because with a large number of tolls it is possible to do better traffic engineering. For small

TABLE 8. Detailed results of best solution found by BRKGA+LS algorithm.

		SPT				SPTF			
Instance	$\mathcal{K}$	**#Path**	**#Hop**	**#Toll**	**#UArc**	**#Path**	**#Hop**	**#Toll**	**#UArc**
SiouxFalls	0	1.97	2.51	-	4.97	1.05	2.14	-	3.24
	10	1.78	2.58	0.21	4.84	1.07	2.10	0.38	3.25
	30	1.32	2.48	1.12	4.00	1.13	2.25	1.09	3.46
	50	1.13	2.28	1.89	3.50	1.09	2.22	1.88	3.34
	70	1.10	2.34	2.96	3.45	1.06	2.20	2.71	3.30
Friedrichshain Center	0	1.96	9.36	-	11.94	1.52	12.24	-	12.82
	10	1.91	9.99	0.21	12.21	1.63	13.63	0.31	13.04
	100	1.42	11.43	1.76	12.58	1.48	12.49	2.14	12.48
	300	1.17	11.50	5.39	12.33	1.00	10.68	5.57	11.68
	500	1.03	10.35	10.39	11.20	1.00	10.44	10.49	11.44
Prenzlauerberg Center	0	2.75	14.72	-	17.11	1.79	18.67	-	18.00
	10	2.61	15.65	0.07	18.02	1.53	16.48	0.38	17.19
	100	1.60	16.41	2.08	17.33	1.30	16.06	2.20	17.02
	450	1.15	15.85	8.07	16.69	1.07	15.86	7.95	16.70
	700	1.12	15.00	15.14	15.97	1.01	14.57	14.69	15.62
Tiergarten Center	0	1.92	15.40	-	17.32	1.12	17.43	-	18.51
	10	2.07	17.20	0.00	18.57	1.07	15.76	0.42	17.23
	100	1.20	16.37	1.01	18.08	1.04	15.98	2.35	17.26
	450	1.03	15.74	8.14	16.98	1.00	16.11	8.95	17.11
	700	1.00	15.81	13.74	16.83	1.00	15.97	15.20	16.97
Mitte Center	0	2.79	14.95	-	17.75	1.33	17.49	-	17.81
	10	3.12	16.81	0.00	18.99	1.38	17.37	0.48	17.60
	100	1.34	15.83	0.45	17.61	1.14	16.13	2.53	16.84
	400	1.05	15.84	6.65	16.77	1.00	15.29	6.58	16.29
	800	1.02	15.12	13.21	16.11	1.02	15.48	14.21	16.32
Anaheim	0	8.71	15.64	-	21.45	1.35	15.67	-	17.46
	10	16.79	19.71	0.00	23.19	1.35	15.16	0.07	16.96
	100	3.29	18.31	0.02	20.61	1.39	15.52	0.74	16.51
	500	1.05	14.78	6.48	15.89	1.04	14.31	6.43	15.45
	800	1.01	14.32	11.85	15.41	1.00	14.50	12.20	15.50
MPF Center	0	5.09	24.35	-	27.54	2.28	31.51	-	29.17
	10	5.25	26.00	0.00	27.95	2.15	30.50	0.23	28.48
	250	3.54	27.98	0.54	29.08	1.34	27.47	2.29	27.37
	1000	1.10	24.98	8.63	25.87	1.21	26.63	9.40	26.33
	2000	1.09	23.79	20.41	24.21	1.00	24.00	20.90	25.00
Barcelona	0	7.37	15.85	-	20.82	1.16	18.73	-	20.73
	10	7.00	18.28	0.01	24.80	1.13	18.70	0.07	20.85
	500	5.02	21.25	0.14	25.03	1.14	19.01	0.69	20.58
	1500	1.38	19.51	8.36	20.35	1.03	18.37	7.21	19.35
	2500	1.08	15.87	16.45	16.77	1.01	16.12	16.69	17.10
Winnipeg	0	3.92	20.72	-	25.96	1.04	24.98	-	25.73
	10	4.22	21.96	0.00	27.15	1.05	24.80	0.15	25.36
	500	3.02	31.04	0.82	32.02	1.17	25.69	2.32	26.11
	1500	1.65	31.76	9.83	25.91	1.11	24.78	9.82	24.84
	2800	1.04	19.77	20.12	20.82	1.01	20.15	20.60	21.14
ChicagoSketch	0	20.04	14.86	-	23.69	1.01	12.30	-	13.32
	10	21.24	15.23	0.00	24.35	1.01	12.19	0.09	13.20
	500	9.88	15.72	0.49	22.49	1.00	12.33	0.95	13.35
	1500	2.64	19.38	4.82	17.57	1.00	12.52	4.97	13.53
	2900	1.16	12.23	12.91	13.22	1.00	11.54	12.23	12.55

values of $\mathcal{K}$ in SPT, the hop count is small because it corresponds to a minimum hop-count path among the zero-cost shortest paths.

The columns #Toll in Table 8 show the average number of tolls that a user traverses on an OD shortest path. Clearly, this value increases with $\mathcal{K}$. Since an increase in $\mathcal{K}$ leads to a decrease in the number of shortest paths (column #Path), the number of distinct arcs (column #UArc) consequently decreases.

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 random-key genetic algorithm was run for this same set of instances.

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 with CPLEX. However, results obtained with the biased random-key genetic algorithm for the complete model shows it has a good 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), FAPERGS (Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul), and PRH PB-217 – Petrobras S.A., Brazil.

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. doi: 10.1007/s10898-010-9527-7.

H. Bar-Gera. Transportation networks test problems, 2013. 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. doi: 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, and M. Thorup. Speeding up dynamic shortest-path algorithms. *INFORMS Journal on Computing*, 20:191–204, 2008.

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. doi: 10.1007/s11590-010-0226-6. URL http://dx.doi.org/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.

J. Ekström, A. Sumalee, and H.K. Lo. Optimizing toll locations and levels using a mixed integer linear approximation approach. *Transportation Research Part B: Methodological*, 46(7):834 – 854, 2012. ISSN 0191-2615. doi: http://dx.doi.org/10.1016/j.trb.2012.02.006. URL http://www.sciencedirect.com/science/article/pii/S0191261512000318.

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. An experimental comparison of biased and unbiased random-key genetic algorithms. *Pesquisa Operacional*, 34:143–164, 2014.

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. Yildirim. A toll pricing framework for traffic assignment problems with elastic demand. In *Transportation and Network Analysis: Current Trends. Miscellanea in honor of Michael Florian*, page 149. Kluwer Academic Publishers, 2002.

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. doi: 10.1007/s11066-008-9024-z. URL http://dx.doi.org/10.1007/s11066-008-9024-z.

J.G. Wardrop. Some theoretical aspects of road traffic research. *Proceedings of the Institution of Civil Engineers, Part II*, 1:325–378, 1952.

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) ISEA TEK, 620 N. Wymore Road, Suite 260, Maitland, FL 32751 USA.
E-mail address: mhirsch@iseatek.com

(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) Network Evolution Research Department, AT&T Labs Research, 200 South Laurel Avenue, Room A5-1F34, Middletown, NJ 07748 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