A biased random-key genetic algorithm for a network pricing problem

Fernando Stefanello, Luciana S. Buriol
Instituto de Informática – Universidade Federal do Rio Grande do Sul
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brazil
`[fstefanello,buriol]@inf.ufrgs.br`

Mauricio G. C. Resende
Internet and Network Systems Research Center, AT&T Labs Research
180 Park Avenue, Room C241, Florham Park, NJ 07932, USA.
`mgcr@research.att.com`

ABSTRACT

Reducing the problems related to transportation networks has challenged researchers from several areas. There are many attempts in solving the different combinatorial network flow problems that arise in this context. In this work, we investigate the problem of applying tolls on some arcs of road networks. The problem of defining tariffs for a given subset of the arcs, maximizing revenue when users take the least cost path, is known as a network pricing problem. In this work we apply a biased random-key genetic algorithm for large instances of this problem. The experimental results reported show that the algorithm found good solutions for large instances in a short time.

KEYWORDS. Bilevel programming, Genetic algorithms, Network pricing problem, Combinatorial Optimization.

1. Introduction

Transportation is an important component on the economy, a promoter of wealth, the development and the welfare of populations. Reducing the problems related to transportation networks has challenged not only traffic engineers, but also researchers from several areas. There are many rules and procedures that are currently in use aiming at improving the traffic flow in a city or road. They can also attend other interests. Motivating the use of bicycle, for example, reduces traffic congestion and improves life style. In spite of the world effort in reducing traffic flow, the still remaining traffic causes congestion in almost all cities and busy roads. Another alternative to reduce some problems is applying tolls on some arcs of the network.

The main idea is that 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.

The optimization of transportation networks using tolls is addressed by many works. The goal of the minimum tollbooth problem (MINTB), first introduced by Hearn and Ramana (1998), is to minimize 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 was proposed. In Stefanello et al. (2013), an extension of Buriol et al. (2010), the authors deal with the problem of locating a fix number $\mathcal{K}$ of tolls, as well as defining their tariffs. For a complete review of road pricing optimization problems we refer the reader to the survey by Tsekeris and Voß (2009).

Another class of problems on flow networks is defined when only a given subset of the arcs can be tariffed. This is the case of the network pricing problem (NPP) introduced by Labbé et al. (????), which is further explored in this work. In an NPP an authority imposes charging tolls in a given set of arcs with the objective of maximizing the revenue, supposing that travellers always take the shortest cost path. The shortest path is computed considering the tolls and a fix link cost. In game theory, a similar problem is known as the Stackelberg game (von Stackelberg, 1952). In this game there is a leader and a follower. The leader plays first choosing the best strategy supposing that the follower reacts in an optimal way to its choice. Knowing the decision of the leader, the follower chooses a strategy considering its own benefit. Similar problems and applications are toll optimization systems (Dewez, 2004), long-distance freight transportation overseas (Brotcorne et al., 2000), airline charging (Castelli et al., 2012), and in telecommunication networks (Başar and Srikant, 2002; Bouhtou et al., 2007).

A bilevel NPP was first introduced by Labbé et al. (????) for a multicommodity network. The problem consists in determining the tariffs to tolls on a subset of arcs of a network, with the objective of maximizing the profit, given that users travel on shortest cost paths. In this problem the authority is supposed to fix its toll prices first, and then the users choose their paths having the complete knowledge of all network costs. In Labbé et al. (????) the general problem was proved to be NP-complete, while particular instances are polynomially solvable, as for example, the single toll arc case. In this work it was also showed how the lower level optimization problem can be replaced by its primal and dual constraints and its optimality conditions, stating that the primal and dual objective functions must be equal, yielding a single level problem which then needs to be linearized. In Roch et al. (2005) the NPP with lower bound constraints on tolls was proved to be strongly NP-hard even for one single commodity. Other results can be found for instance in Bouhtou et al. (2007), Dewez (2004), Heilporn et al. (2010) and Brotcorne et al. (2012).

In this paper, we approach the multicommodity network pricing problem. The objective is to determine the tariff of a given subset of arcs of the network maximizing the revenue computed by travellers that choose their shortest cost path routes. This work is a preliminary study which proposes a biased random-key genetic algorithm (BRKGA) to solve large scale instances from the bilevel multicommodity network pricing problem. We further present a set of experiments, providing some conclusions and directions to future work.

This paper is organized as follows. In Section 2 we present a overview and a mathematical formulation of the network pricing problem. The biased random-key genetic algorithm to solve this problem is presented in Section 3. Computational results are reported in Section 4. Finally, conclusions are drawn in Section 5.

2. The network pricing problem

Consider a network represented by a directed graph $G = (V, A, c)$ where V represents the set of nodes (i.e., vertices or points of interest), and A the set of arcs (i.e., links or road segments). The set A is partitioned into two subsets, i.e., $A = A_1 \cup A_2$. Subset A_1 contains the $\mathcal{K} = |A_1|$ arcs that can be tariffed, and belongs to the leader, while A_2 is owned by other agent in the network and the arc costs c_a are known *a priori*. Besides the tariffs, arcs from $\mathcal{K} = |A_1|$ also has a cost c_a. Thus, the arcs belonging to A_1 have cost $c_a + t_a$, while the arcs that belong to A_2 have cost c_a, where c_a is the fix cost of the arc and t_a is the tariff applied to the arc a.

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 $k = 1, \ldots, |K|$ has an associated demand of traffic flow d_k, i.e., for each OD pair $(o(k), d(k))$, there is an demand d_k that emanates from node $o(k)$ and terminates in node $d(k)$.

To ensure that the problem is bounded, it is assumed that for each commodity $k \in K$ there is an upper bound on the amount the customer is willing to pay, or there exists a path from source to destination which uses only fixed cost arcs $a \in A_2$.

The multicommodity NPP has been formulated as follows (Labbé et al., ????):

$$\max \sum_{a\in A_1} t_a \sum_{k\in K} x_a^k \tag{1}$$

$$\min \sum_{k\in K} \left\{ \sum_{a\in A_1} (c_a + t_a) x_a^k + \sum_{a\in A_2} c_a x_a^k \right\} \tag{2}$$

$$\sum_{a\in IN(v)} x_a^k - \sum_{a\in OUT(v)} x_a^k = \begin{cases} -d_k, & \text{if } v = d(k) \\ d_k, & \text{if } v = o(k) \\ 0, & \text{otherwise} \end{cases} \qquad \forall v \in V,\ \forall k \in K, \tag{3}$$

$$x_a^k, t_{a'} \in \Re^+ \qquad \forall a' \in A_1,\ \forall a \in A,\ \forall k \in K \qquad . \tag{4}$$

Here $IN(v)$ is the set of arcs entering v, and $OUT(v)$ is the set of arcs leaving v. In this model, the variables x_a^k represent the flow of commodity $k \in K$ on arc $a \in A$.

This model is a bilinear bilevel program, since the upper level is linear in the tariff variables and the lower level is linear in the arc choice variables (van Hoesel, 2008), resulting in a non linear formulation in the combination of these variables.

Observe that once the tariffs are defined, one can easily choose among all s-t paths of minimum total cost one path that maximizes the revenue. In other words, we assume that the follower always makes the best choice for the leader. A naturally extension of the NPP is to allow negative tolls. In this case, the values are incentives to travellers take this routes. In this work we limit only to non-negative tolls.

A mixed integer linear formulation is provided for this problem in Labbé et al. (????). This model was obtained by replacing the lower level problem by its optimality conditions. Thus, an arc formulation for NPP has been formulated as follows:

$$\max \sum_{k\in K} \sum_{a\in A_1} d_k t_a^k \tag{5}$$

subject to

$$\lambda_i^k - \lambda_j^k \le c_a + T_a \qquad \forall a = (i,j) \in A_1, \forall k \in K \tag{6}$$

$$\lambda_i^k - \lambda_j^k \le c_a \qquad \forall a = (i,j) \in A_2, \forall k \in K \tag{7}$$

$$\sum_{a\in A_1} \left(c_a x_a^k + t_a^k\right) + \sum_{a\in A_2} c_a x_a^k = \lambda_{d(k)}^k - \lambda_{o(k)}^k \qquad \forall k \in K \tag{8}$$

$$\sum_{a\in IN(v)} x_a^k - \sum_{a\in OUT(v)} x_a^k = \begin{cases} -1, & \text{if } v = d(k) \\ 1, & \text{if } v = o(k) \\ 0, & \text{otherwise} \end{cases} \qquad \forall v \in N, \forall k \in K \tag{9}$$

$$-Mx_a^k \le t_a^k \le Mx_a^k \qquad \forall a \in A_1, \forall k \in K \tag{10}$$

$$-M\left(1 - x_a^k\right) \le t_a^k - T_a \le M\left(1 - x_a^k\right) \qquad \forall a \in A_1, \forall k \in K \tag{11}$$

$$x_a^k \in \{0,1\}, t_a^k, T_a \in \Re^+ \qquad \forall a \in A_1, \forall k \in K \tag{12}$$

$$x_a^k \ge 0 \qquad \forall a \in A_2, \forall k \in K \tag{13}$$

$$\lambda_v^k \ge 0 \qquad \forall v \in V, \forall k \in K \tag{14}$$

Objective function (5) maximizes the revenue of the demand of each commodity that traverses the tarrifed arcs. Constraint set (6) and (7) defines the weight distance from the vertices. Constraint set (8) helps to define the tariffs of the tarrifed arcs based on the vertice distances. Constraint set (9) guarantees flow conservation. Constraint set (10) and (11) set the tariffs only on arcs with flow, and the last constraint sets define the domain of the variables (12)–(14) .

This model can be improved in the case of having many commodities with a same destination. The set of variables $\lambda_v^k, \forall\ v \in V$ and $\forall k \in K$ can be replaced by a set of variables $\lambda_v^q, \forall v \in V$ and $\forall q \in Q$ where Q is the set of all destination nodes. For a large number of commodities $|Q| \ll |K|$ this modification reduces the number of variables in comparison to the original model.

3. A biased random-key genetic algorithm

In this section we briefly describe the biased random-key genetic algorithm (RKGA) proposed to solve the NPP. 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. 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 through *generations*.

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 usually $p - p_e > p_e$, in this case $Pr(s) > Pr(\bar{s})$. 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 k-vector of random keys, where k=$|\mathcal{K}|$, the cardinality of the tariffed arcs (A_1) in the network. Each random key corresponds to a tariff. Each arc $a \in \mathcal{K}$ has a tariff in the interval $[0, t_{\max}]$. Let d_k^0 the distance from node s to node t of the commodity k when the tariffed arcs are defined to zero and d_k^∞ the distance from s to t for commodity k when the tariffed arcs are defined to a large value. Thus, a natural upper bound $t_{\max}$ on the value of a tariff on the network is given by

$$t_{max} = \max_{k \in K} \left\{ d_k^\infty - d_k^0 \right\}.$$

Two kinds of initial solutions are generated. The model (6)-(14) was solved with constraint (12) relaxed. This generates one feasible solution. We observed that the values of tariffs of this solution are, in most cases, an upper approximation of the tariffs of the optimal solution. All other solutions are randomly generated. We use this upper bound to limit the tariff values and for providing an initial solution for the BRKGA. Each random solution is then generated by setting the keys to a value up to the value found in the solution generated by the relaxed model.

Demands are routed forward to their destinations on shortest weight paths. Observe that tariffed links have weights equal to their fixed cost plus their tariffs ($c_a + t_a$) and untariffed links have only a fixed cost (c_a). For each commodity, all paths of minimum cost are evaluated and the demand is sent by the higher cost path. In the case of a tie, the path with the higher number of tariffed arcs is selected. Finally, in case of a second tie, the tie is break using the outgoing arc with the lowest index. Once the demands are routed, the revenue of each tariff can be computed for all commodities. The solution fitness value is then calculated.

Two additional features are implemented to maintaining the diversity of the population. First, at each generation, if two individuals have the same fitness value we apply a perturbation on each of them. A value $\delta \in [-d, d]$ where $d = \lceil t_{max} * 0.1 \rceil$ is added to each tariff value. Note that the key value limitation between $[1, t_{max}]$ should be respected.

Moreover, the population is restarted after 50 generations without improvement of the best solution. This is done by replacing all individuals (except the one with best fitness) by random generated individuals, as it was done in the construction of the initial population.

4. Computational results

In this section we present computational experiments with the models and algorithms presented in the previous sections. Initially, we describe the dataset used in the experiments. Then, we detail some experiments with CPLEX applied on the mathematical model (5)–(14) and a components based on the relaxation of this model added to a basic implementation of the BRKGA that help to improve the results. Finally, preliminary results for the BRKGA are reported with some considerations.

The experiments were done on a computer with an Intel Core i5 2300 processor running at 2.80 GHz, with 4 GB of DDR3 RAM of main memory, and Ubuntu 12.10 Linux operating system. The BRKGA was implemented in C++ and compiled with the g++ compiler, version 4.7.2. We used CPLEX 12.4[1] (API C++) with default configuration.

Three types of networks are used in the first experiment, as show in Figure 1.

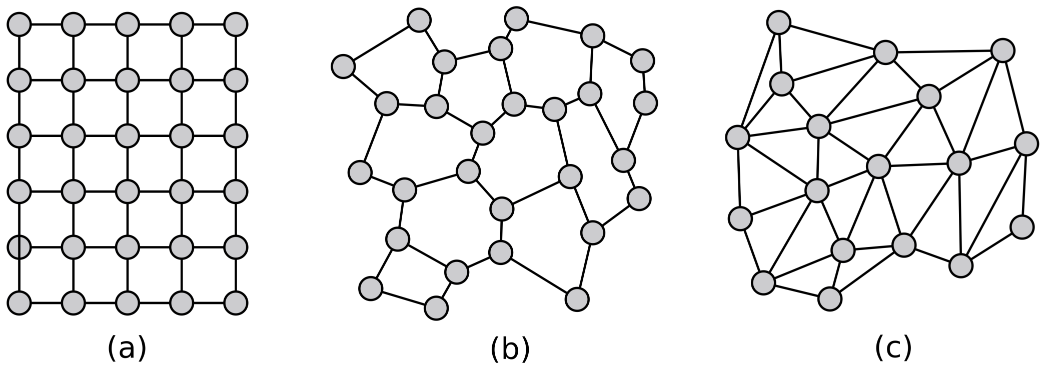

Figure 1. Network structures: (a) Grid network; (b) Delaunay network; (c) Voronoi network.

For each edge from the original structure two directed arcs with opposite directions were created. A random weight $c_a \in [1, c_{max}]$ is assigned to each one. Distinct origin and destination nodes of each commodities are also randomly chosen, as well as the demand is randomly generated in the interval $[1, d_{max}]$ (by default $d_{max} = 20$).

Tariffed arcs are selected based on the description provide on Brotcorne et al. (2000). However, some changes were made in order to increase the number of travellers through tarrifed arcs for increasing the difficulty to solve the problem. Shortest paths are then calculated. The frequency that each arc belongs to a shortest path is computed. The first $|A_1|/2$ arcs are selected considering the decreasing order of frequencies and the weights of these arcs are increased by a large value (usually a maximum integer representation). Their frequency count are also increased by a large value (it was used $|A|$). Next, considering the new values of weights, the demands are routed again and the frequencies are updated. Finally in a decreasing order of frequencies, the tarrifed arcs are selected. Once a arc is tarrifed, a test is performed to check if there is at least one untariffed path from each commodity. If there is no such a path, the toll is removed and the next arc is selected to receive the toll. This process is repeated until $\mathcal{K}$ tariffs are assigned.

[1] www-01.ibm.com/software/integration/optimization/cplex-optimizer

Sets of instances are created with 200, 500 and 1000 arcs (the exact number of arcs can vary according to the network structure). The fixed maximum weight of arcs c_{max} is defined to10 and 30. The number of origin-destination pairs is selected from 50, 200 and 500, and the proportion of tariffed arcs from 10% and 20%, generating a total of 108 instances[2].

4.1. Results for the mathematical model

This section reports experiments with CPLEX running the model (5)–(14) considering the subset of instances with 200 arcs from Table 1.

The columns present the name of the instances, the CPLEX gap and the computational time (limited to 1h run). The name of each instance is composed by the network structure, #arcs, #o-d pairs, and the max arc weight d_{max}. A '-' in the table indicates the cases in which the computer memory was exceeded and then the run was interrupted.

Table 1. Computational results for the mathematical model.

	$\mathcal{K} = 10\%$		$\mathcal{K} = 20\%$	
Instance	**gap**	**Time(s)**	**gap**	**Time(s)**
Grid-0200-0050-10	0,00	9,94	3,46	3.600,23
Grid-0200-0050-30	0,00	2,69	0,00	126,57
Grid-0200-0200-10	-	-	200,09	3.600,66
Grid-0200-0200-30	54,24	3.600,32	286,44	3.600,98
Grid-0200-0500-10	451,26	3.601,28	366,22	3.600,13
Grid-0200-0500-30	734,92	3.600,06	1.974,75	3.600,08
Delaunay-0200-0050-10	0,00	1,44	0,00	2,88
Delaunay-0200-0050-30	0,00	11,33	0,00	105,05
Delaunay-0200-0200-10	0,00	104,75	13,41	3.606,50
Delaunay-0200-0200-30	1,74	3.601,21	11,37	3.605,06
Delaunay-0200-0500-10	118,84	3.602,50	731,74	3.600,06
Delaunay-0200-0500-30	-	-	604,50	3.601,28
Voronoi-0200-0050-10	0,00	1,57	0,00	169,75
Voronoi-0200-0050-30	0,00	6,67	0,00	1.099,20
Voronoi-0200-0200-10	5,08	3.600,49	117,19	3.600,79
Voronoi-0200-0200-30	3,99	3.600,39	92,02	3.600,58
Voronoi-0200-0500-10	-	-	-	-
Voronoi-0200-0500-30	-	-	-	-

As expected, Table 1 shows that a higher number of tarrifed arcs increases the difficulty of the solver to prove optimality. The same is caused by the increase of the number of commodities. This occurs because the variables of this model are directly related to these parameters.

A relaxed version of the model was also solved. Despite the relaxation, the solver provides fractional values of tariffs. The revenue is computed considering these tariffs. The solution value is about 40% less of the optimal values found in the previous experiment. The relaxed solutions are on average better than a random generated solution. In a set of

[2]Available at inf.ufrgs.br/~fstefanello

1000 random solutions for each instance, the average gap quality is around 64% less than the quality of the relaxed solution. Related to the best of 1000 random solutions, the gap is still 20% less of the quality of the relaxed solution.

Analyzing the results, it was observed that for almost all arcs the tariffs are higher or equal than the optimal tariff values. Comparing the values of tariffs on the relaxed solution with the tariffs of the optimal solutions obtained in the previous experiment, 42.5% of the values of tariffs are equal to the optimal solution, and 43.66% have higher values, while only 13.82% are smaller values. Furthermore, the average of difference between the values is 3.21 units, meaning that the obtained tariff values are close to the optimal ones. Thus, these values can be used as an upper bound for the tariff values, providing a good approximation. The time to compute this solution is less than 2 seconds for the instances from Table 1. This represents less than 3% of the BRKGA time reported in the next section.

4.2. Results from the biased random-key genetic algorithm

This section shows results obtained with the biased random-key genetic algorithm applied to a larger set of large scale instances of the NPP.

The experiments with the BRKGA were done with a population size of $p = 50$, an elite set of size $p_e = 0.25p$, a mutant set of size $p_m = 0.05p$, and an elite key inheritance probability of $\rho_A = 0.7$.

Table 2 shows the results obtained, averaged over ten runs with different random seeds and using as stopping criterium 2000 generations of the BRKGA. In this table we report the best know solution value (Best), found by the CPLEX in the previous experiment (presented in bold if is optimal) or an extended execution of the BRKGA to 3000 generations. The columns Relax and Time(s) report the CPLEX results for solving the relaxed model (5)–(14). The column Relax shows the total revenue calculated with the tariffs obtained by the solver. Since the tolls values in the relaxed version can be non-integer, the revenue can also be fractional. The column Time(s) show the running time of the solver in second. The columns Avg, Max, SD and Time(s) report respectively the values of the average revenue, best revenue, standard deviation and the computational time in seconds for the BRKGA. The reported running time is not cumulative with the running time of the relaxed model. Finally, we report results for the cases where 10% and 20% of the arcs are tariffed.

Table 2. Computational results for the BRKGA.

	$\mathcal{K} = 10\%$							$\mathcal{K} = 20\%$						
Instance	**Best**	**Relax**	**Time(s)**	**Avg**	**Max**	**SD**	**Time(s)**	**Best**	**Relax**	**Time(s)**	**Avg**	**Max**	**SD**	**Time(s)**
Delaunay-0200-0050-10	**747**	652.0	0.1	747.0	**747**	0.0	39.4	**1473**	1,090.0	0.1	1,472.6	**1,473**	1.3	37.7
Delaunay-0200-0050-30	**1814**	1,290.3	0.1	1,799.0	1,807	5.7	37.4	**3144**	2,526.0	0.1	2,985.3	**3,114**	82.4	38.1
Delaunay-0200-0200-10	**2236**	1,666.0	0.5	2,236.0	**2,236**	0.0	86.4	3339	2,179.4	0.7	3,270.6	3,331	30.2	83.9
Delaunay-0200-0200-30	6451	4,560.0	0.7	6,388.6	6,451	47.4	76.7	14001	8,293.9	0.8	13,612.4	13,876	148.1	75.4
Delaunay-0200-0500-10	6473	3,722.6	2.7	6,473.0	6,473	0.0	169.7	9599	6,125.5	2.6	9,432.4	9,599	111.2	153.5
Delaunay-0200-0500-30	15507	8,500.8	2.8	15,419.4	15,492	52.1	143.9	24540	14,621.5	4.3	23,799.0	24,540	457.2	142.4
Delaunay-0500-0050-10	1227	1,008.0	0.4	1,222.1	1,227	4.5	105.4	2048	1,710.0	0.5	2,028.9	2,048	12.1	109.3
Delaunay-0500-0050-30	3849	2,224.4	0.5	3,756.4	3,847	52.2	94.5	4774	3,457.0	0.6	4,611.0	4,774	146.3	96.6
Delaunay-0500-0200-10	3089	2,160.5	3.1	3,058.5	3,089	12.8	241.4	6495	4,304.7	4.0	6,318.0	6,430	99.8	237.0
Delaunay-0500-0200-30	11961	8,044.6	3.5	11,786.6	11,961	101.1	219.5	19756	12,595.3	3.9	19,199.4	19,663	208.0	220.4
Delaunay-0500-0500-10	7354	4,446.9	9.3	7,256.6	7,343	69.1	354.6	13980	8,046.2	11.0	13,546.2	13,861	153.5	358.4
Delaunay-0500-0500-30	26640	13,822.5	10.4	26,167.9	26,619	432.9	334.9	39004	21,382.8	10.8	37,988.9	38,890	586.6	327.4
Delaunay-1000-0050-10	1696	1,239.4	1.5	1,688.3	1,696	6.3	232.2	2388	1,971.0	2.1	2,359.5	2,383	15.8	237.0
Delaunay-1000-0050-30	4410	3,277.0	1.6	4,309.8	4,407	72.0	216.4	7580	6,050.0	3.1	7,417.1	7,515	69.4	218.8
Delaunay-1000-0200-10	5601	3,763.5	7.0	5,536.4	5,600	51.7	654.0	9155	6,300.9	11.5	8,964.6	9,102	104.8	646.4
Delaunay-1000-0200-30	15018	8,189.4	11.5	14,716.6	14,949	158.9	609.3	29476	19,014.3	13.2	28,696.3	29,389	478.2	612.4
Delaunay-1000-0500-10	16612	9,064.0	24.2	16,290.5	16,486	222.2	943.3	25151	13,970.9	36.2	24,656.9	25,149	304.7	939.9
Delaunay-1000-0500-30	25710	16,614.3	27.2	25,373.7	25,685	269.1	907.1	61565	34,601.9	35.9	60,098.7	61,015	551.0	895.9
Grid-0200-0050-10	**3893**	2,227.0	0.2	3,893.0	**3,893**	0.0	51.2	**5525**	4,094.1	0.2	5,313.7	5,441	81.8	52.1
Grid-0200-0050-30	**6543**	4,508.1	0.2	6,496.7	6,501	2.9	45.6	14151	10,704.0	0.2	13,694.5	14,012	162.9	49.2
Grid-0200-0200-10	10454	5,537.4	1.0	10,407.9	10,454	39.5	109.6	15898	8,470.5	1.3	15,620.3	15,898	252.1	110.7
Grid-0200-0200-30	23760	13,129.1	1.0	23,065.9	23,672	421.9	99.4	49638	25,584.2	1.3	48,745.1	49,638	1,054.4	101.8
Grid-0200-0500-10	18945	10,586.6	2.9	18,825.5	18,945	82.3	206.1	34367	17,619.1	3.3	33,905.8	34,342	259.2	203.2
Grid-0200-0500-30	56834	27,783.2	3.7	55,965.1	56,580	614.8	186.6	94813	41,060.2	4.4	91,049.2	93,937	1,869.4	192.4
Grid-0500-0050-10	6779	3,834.0	1.1	6,711.6	6,779	54.6	133.8	10103	5,395.2	1.7	9,719.6	10,065	316.9	136.2
Grid-0500-0050-30	7223	4,312.0	0.9	7,099.1	7,211	76.7	123.7	16499	13,211.4	1.3	15,890.0	16,370	363.3	126.1
Grid-0500-0200-10	11227	6,321.8	4.8	11,067.3	11,213	72.4	344.9	23527	11,657.4	4.2	23,002.8	23,450	368.3	355.7
Grid-0500-0200-30	40486	21,479.2	4.8	38,827.3	40,301	913.1	335.8	75548	32,123.7	5.9	73,920.6	75,464	1,241.8	332.3
Grid-0500-0500-10	17662	8,457.0	14.2	17,491.9	17,636	109.5	539.1	54182	18,837.3	16.2	52,598.8	53,999	1,041.1	601.5
Grid-0500-0500-30	79060	36,411.6	14.1	77,912.9	79,007	795.7	524.9	129323	57,971.3	21.7	125,633.8	128,804	2,106.5	534.4
Grid-1000-0050-10	5612	4,040.0	4.1	5,508.6	5,608	55.3	276.2	11697	8,743.0	7.3	11,336.8	11,665	150.5	277.5
Grid-1000-0050-30	12731	8,129.0	4.7	12,342.1	12,604	280.6	264.4	24309	20,187.3	9.5	22,152.5	23,716	796.9	300.7
Grid-1000-0200-10	20265	8,740.1	15.5	20,008.7	20,237	158.4	865.8	37919	16,049.5	25.3	36,895.8	37,300	298.8	895.9
Grid-1000-0200-30	53043	20,330.4	13.9	51,180.6	53,012	939.0	840.3	91484	47,980.6	24.0	87,838.7	91,253	2,780.3	828.7
Grid-1000-0500-10	40893	13,270.9	54.0	40,353.4	40,844	406.6	1,497.0	71311	26,301.5	76.2	69,605.2	70,894	753.4	1,501.9
Grid-1000-0500-30	80735	33,753.0	45.3	79,056.6	80,327	1,108.0	1,369.6	192204	71,360.8	66.3	185,104.5	190,730	3,046.9	1,363.6
Voronoi-0200-0050-10	**1559**	1,263.8	0.2	1,559.0	**1,559**	0.0	58.1	**3643**	2,799.0	0.2	3,391.4	3,511	76.9	60.5
Voronoi-0200-0050-30	**4259**	2,896.1	0.2	4,149.2	4,204	27.1	54.8	**8550**	7,251.5	0.2	7,980.6	8,333	163.5	57.9
Voronoi-0200-0200-10	5563	3,621.4	1.0	5,518.9	5,563	40.7	130.6	11535	7,423.3	1.4	11,423.1	11,535	91.2	134.5
Voronoi-0200-0200-30	20048	11,523.0	1.1	19,826.0	20,048	290.8	120.4	37669	23,367.5	1.5	37,009.2	37,665	506.6	119.0
Voronoi-0200-0500-10	11002	7,248.7	3.4	10,785.7	11,002	183.4	195.2	42877	21,236.2	3.5	41,801.0	42,546	763.4	206.3
Voronoi-0200-0500-30	45009	22,606.0	3.9	44,417.0	44,987	617.3	195.8	80054	42,007.3	4.3	78,602.0	79,849	638.9	198.3
Voronoi-0500-0050-10	4064	2,949.2	1.3	4,001.2	4,064	29.5	137.2	8315	5,663.6	1.6	8,116.0	8,290	85.6	141.4
Voronoi-0500-0050-30	9144	5,500.7	1.1	8,785.9	9,089	215.3	157.3	21317	19,580.0	1.7	21,100.9	21,313	156.0	164.0
Voronoi-0500-0200-10	7940	4,599.3	5.3	7,828.0	7,940	84.6	416.4	18080	12,411.0	5.0	17,610.7	18,033	266.9	421.4
Voronoi-0500-0200-30	33699	18,590.5	4.6	32,539.8	33,547	640.0	411.5	72276	44,523.4	6.6	70,345.8	71,915	1,226.7	417.1
Voronoi-0500-0500-10	20377	10,262.8	14.3	19,927.3	20,244	252.7	681.1	48475	24,588.8	18.6	47,365.8	48,111	597.1	702.6
Voronoi-0500-0500-30	75253	32,314.0	17.5	73,406.3	74,777	943.8	674.9	138913	55,682.3	23.0	135,419.4	137,742	1,753.8	674.1
Voronoi-1000-0050-10	4795	3,492.3	2.6	4,737.8	4,784	22.7	320.5	10806	9,382.0	4.6	10,721.7	10,777	39.7	337.9
Voronoi-1000-0050-30	11162	9,787.0	3.2	11,071.0	11,161	91.9	322.7	22135	18,066.0	7.2	21,183.6	22,119	650.4	335.5
Voronoi-1000-0200-10	12014	7,579.0	15.6	11,496.6	11,984	261.1	985.8	31665	20,183.9	28.2	30,933.8	31,490	348.8	998.1
Voronoi-1000-0200-30	40808	26,114.4	15.2	39,592.2	40,488	510.3	1,038.8	71007	44,108.7	21.5	68,606.4	70,347	821.8	1,050.3
Voronoi-1000-0500-10	32399	15,755.9	48.6	31,902.9	32,299	354.3	1,869.8	83231	44,191.7	66.9	81,426.4	82,926	1,093.8	1,934.6
Voronoi-1000-0500-30	88483	38,070.3	61.1	85,783.4	88,051	1,625.9	1,827.7	226550	146,376.1	65.2	218,943.5	223,702	2,974.4	1,890.9

For the small instances, the BRKGA found the optimal solution or slightly less revenue than the optimal solution value. This indicates that at least for this set of instances, the proposed algorithm has a good performance and we expect similar behavior for the other sets of instances. Related to the network structure, we observed that the grid networks have an average of standard deviation and running times slightly worse than the other structures. This occurs because in this kind of network structures the path length for each commodity tends to be higher than in Voronoi and Delaunay networks. By the same reason, in this kind of structures the revenue tends to be higher.

In the instances with weights c_a between $[1, 30]$ we observed that the standard deviation is worse than in the case with weight between $[1, 10]$. Naturally this behavior is expected because in the first case a higher variation of the tariff values and revenue are observed.

The running times of the algorithm is sensible to the number of arcs, commodities and slightly by the network structure. On the other hand, is not sensible to other parameters, specially to the percentage of tarrifed arcs. We also observed that the computational times are acceptable considering the size of the instances.

5. Conclusions

In this paper we presented a preliminary study of the network pricing problem. A biased random-key genetic algorithm is proposed to solve the problem and a set of experiments was performed in a set of diverse and large scale network instances.

In the experiments we observed that the relaxation of the arc formulation mathematical model solved by CPLEX provides a hight quality initial solution and a good approximation for an upper bound of the tariff values. Furthermore, CPLEX was not able to solve the mixed integer version of this model for the large scale networks instances, motivating the use of heuristics for solving the problem.

This first version of BRKGA shows a good performance of the algorithm, reaching the optimal solution or a good approximation of the best revenue. We also evaluate some characteristics and behaviors of the proposed algorithm on the tested instances.

Finally, as future works we aim at introducing a local search procedure to the BRKGA. Moreover, we intend to use exact methods to solve subproblems as an intensification criteria.

Acknowledgements

This work has been partially supported by CAPES (Coordenação de Aperfeiçoamento de Pessoal de Nível Superior) - Brazil, FAPERGS (Fundação de Amparo à Pesquisa do Estado do Rio Grande do Sul) and PRH PB-17 – Petrobras S.A., Brazil.

References

Bai, L., Hearn, D., Lawphongpanich, S. (2010). A heuristic method for the minimum toll booth problem. Journal of Global Optimization 48, 533–548.

Başar, T., Srikant, R. (2002). A Stackelberg Network Game with a Large Number of Followers. Journal of Optimization Theory and Applications 115 (3), 479–490.

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

Beckmann, M., McGuire, C., Winsten, C. (1956). Studies in the economics of transportation. Yale University Press.

Bouhtou, M., van Hoesel, S., van der Kraaij, A. F., Lutton, J.-L. (Jan. 2007). Tariff Optimization in Networks. INFORMS Journal on Computing 19 (3), 458–469.

Brotcorne, L., Cirinei, F., Marcotte, P., Savard, G. (Nov. 2012). A Tabu search algorithm for the network pricing problem. Computers & Operations Research 39 (11), 2603–2611.

Brotcorne, L., Labbé, M., Marcotte, P., Savard, G. (Aug. 2000). A Bilevel Model and Solution Algorithm for a Freight Tariff-Setting Problem. Transportation Science 34 (3), 289–302.

Buriol, L., Hirsch, M., Pardalos, P., Querido, T., Resende, M., Ritt, M. (2010). A biased random-key genetic algorithm for road congestion minimization. Optimization Letters 4, 619–633.

Castelli, L., Labbé, M., Violin, A. (Jun. 2012). A Network Pricing Formulation for the revenue maximization of European Air Navigation Service Providers. Transportation Research Part C: Emerging Technologies.

Dewez, S. (2004). On the toll setting problem. Ph.D. thesis.

Gonçalves, J., Resende, M. (2011). Biased random-key genetic algorithms for combinatorial optimization. J. of Heuristics 17, 487–525.

Gonçalves, J., Resende, M., TOso, R. (December 2012). Biased and unbiased random-key genetic algorithms: An experimental analysis. Tech. rep., AT&T Labs Research, Florham Park, New Jersey.

Hearn, D. W., Ramana, M. V. (1998). Solving congestion toll pricing models. Equilibrium and Advanced Transportation Modeling, 109–124.

Heilporn, G., Labbé, M., Marcotte, P., Savard, G. (2010). A polyhedral study of the network pricing problem with connected toll arcs. Networks 55 (3), 234–246.

Labbé, M., Marcotte, P., Savard, G. (????). Management Science.

Roch, S., Savard, G., Marcotte, P. (Aug. 2005). An approximation algorithm for Stackelberg network pricing. Networks 46 (1), 57–67.

Spears, W. M., DeJong, K. A. (1991). On the virtues of parameterized uniform crossover. In: Proc. of the Fourth International Conference on Genetic Algorithms. pp. 230–236.

Stefanello, F., Buriol, L. S., Hirsch, M. J., Pardalos, P. M., Querido, T., Resende, M. G. C., Ritt, M. (january 2013). On the minimization of traffic congestion in road networks with tolls, preprint submitted to European Journal of Operational Research.

Tsekeris, T., Voß, S. (2009). Design and evaluation of road pricing: state-of-the-art and methodological advances. Netnomics 10, 5–52, 10.1007/s11066-008-9024-z.

van Hoesel, S. (Sep. 2008). An overview of Stackelberg pricing in networks. European Journal of Operational Research 189 (3), 1393–1402.

von Stackelberg, H. (1952). The theory of the market economy. Oxford University Press, Oxford, England.

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