
Path-based formulations of a bilevel toll setting problem

Mohamed Didi-Biha¹, Patrice Marcotte² and Gilles Savard³

¹ Laboratoire d'Analyse non linéaire et Géométrie, Université d'Avignon et des Pays de Vaucluse, Avignon, France mohamed.didi-biha@univ-avignon.fr

² CRT, Département d'Informatique et de Recherche Opérationnelle, Université de Montréal, Montréal (QC), Canada marcotte@iro.umontreal.ca

³ GERAD, Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal, Montréal (QC), Canada gilles.savard@polymtl.ca

Summary. A version of the toll setting problem consists in determining profit maximizing tolls on a subset of arcs of a transportation network, given that users travel on shortest paths. This yields a bilevel program for which we propose efficient algorithms based on path generation.

Key words. Pricing. Bilevel programming. Networks. Column generation. Combinatorial optimization.

1 Introduction

Bilevel programming offers a convenient framework for the modelling of pricing problems, as it allows to take explicitly into account user behaviour. One of the simplest instances was analyzed by Labbé et al. [8], who considered a toll optimization problem (TOP) defined over a congestion-free, multicommodity transportation network. In this setting, a highway authority (the “leader”) sets tolls on a subset of arcs of the network, while the users (the “follower”) assign themselves to shortest⁴ paths linking their respective origin and destination nodes. The goal of the leader being to maximize toll revenue, it is not in its interest to set tolls at very high values, in which case the users will be discouraged from using the tolled subnetwork. The problem, which consists in striking the right balance between tolls that generate high revenues and tolls that attract customers, can be formulated as a combinatorial program that subsumes NP-hard problems, such as the Traveling Salesman Problem

⁴ It is assumed that costs and travel times are expressed in a common unit, i.e., the monetary perception of one unit of travel time is uniform throughout the user population.

(see Marcotte et al [11] for a reduction). Following the initial NP-hardness proof by Labbé et al., complexity and approximation results have also been obtained by Roch et al. [12] and Grigoriev et al. [5].

The aim of the present work is to assess the numerical performance of path-based reformulations of TOP, and to show their ability to solve to optimality medium-sized instances, and to near optimality large-scale instances. This stands in contrast with arc-based methods that have been proposed by Labbé et al. [8] and Brotcorne et al. [1]. Note that Bouhtou et al. [2] have recently proposed, together with arc-based methods, a path-based approach operating on a compact reformulation of the problem.

The structure of the paper is as follows: Section 2 introduces three Mixed Integer Programming (MIP) formulations for TOP; Section 3 introduces a path generation framework; Section 4 details a sequential implementation; Section 5 presents numerical results achieved on randomly generated test problems; Section 6 concludes with avenues for further research.

2 A bilevel formulation

In this section, we present three MIP formulations of TOP. The first, initially proposed by Labbé et al. [8], relies on the optimality conditions associated with an arc-commodity formulation. The second utilizes both arc and path variables, while the third is entirely path-based.

TOP can be analyzed as a leader-follower game that takes place on a multicommodity network $G = (K, N, A)$ defined by a set of origin-destination couples K , a node set N and an arc set A . The latter is partitioned into the subset A_1 of toll arcs and the complementary subset A_2 of toll-free arcs. We endow each arc $a \in A$ with a fixed travel delay c_a . Toll arcs $a \in A_1$ also involve a toll component t_a , to be determined, that is expressed in time units, for the sake of consistency. The demand side is represented by numbers n^k denoting the demand for travel between the origin node $o(k)$ and the destination $d(k)$ associated with commodity $k \in K$. With each commodity is associated a demand vector b^k whose components are, for every node i of the network:

$$b_i^k = \begin{cases} -n^k & \text{if } i = o(k), \\ n^k & \text{if } i = d(k), \\ 0 & \text{otherwise.} \end{cases}$$

Letting x_a^k denote the set of commodity flows and i^+ (respectively i^-) the set of arcs having i as their head node (respectively tail node), TOP can be formulated as a bilevel program involving bilinear objectives at both decision levels:

$$\begin{aligned}
 \text{TOP:} \quad & \max_{t,x} \sum_{k \in K} \sum_{a \in A_1} t_a x_a^k \\
 \text{subject to} \quad & t_a \leq t_a^{\max} \quad \forall a \in A_1 \\
 & \forall k \in K \left\{ \begin{array}{l} x^k \in \arg \min_{\bar{x}} \sum_{a \in A_1} (c_a + t_a) \bar{x}_a + \sum_{a \in A_2} c_a \bar{x}_a \\ \text{subject to} \quad \sum_{a \in i^-} \bar{x}_a - \sum_{a \in i^+} \bar{x}_a = b_i^k \quad \forall i \in N \\ \bar{x}_a \geq 0 \quad \forall a \in A. \end{array} \right.
 \end{aligned}$$

In the above formulation, the leader controls both the toll and flow variables. However the lower level ‘argmin’ constraint forces the leader to assign flows to shortest paths with respect to the current toll levels. In order to prevent the occurrence of trivial situations, the following conditions are assumed to hold throughout the paper:

1. There does not exist a profitable toll vector that induces a negative cost (delay) cycle in the network. This condition is clearly satisfied if all delays c_a are nonnegative.
2. For each commodity, there exists at least one path composed solely of toll-free arcs.

Under the above assumptions, the lower level optimal solution corresponds to a set of shortest paths, and the leader’s profit is bounded from above.

A single-level reformulation of TOP is readily obtained by replacing the lower level program by its primal-dual optimality conditions. If one expresses the latter by the equality of the primal and dual objectives, we obtain the nonlinearly-constrained program

$$\begin{aligned}
 \text{MIP:} \quad & \max_{t,x,\lambda} \sum_{k \in K} \sum_{a \in A_1} t_a x_a^k \\
 \text{subject to} \quad & t_a \leq t_a^{\max} \quad \forall a \in A_1 \\
 & \forall k \in K \left\{ \begin{array}{l} \sum_{a \in i^-} x_a^k - \sum_{a \in i^+} x_a^k = b_i^k \quad \forall i \in N \\ \lambda_j^k - \lambda_i^k \leq c_a + t_a \quad \forall a = (i, j) \in A_1 \\ \lambda_j^k - \lambda_i^k \leq c_a \quad \forall a \in A_2 \\ \sum_{a \in A_1} (c_a + t_a) x_a^k + \sum_{a \in A_2} c_a x_a^k = (\lambda_{o(k)}^k - \lambda_{d(k)}^k) n^k \\ x_a^k \geq 0 \quad \forall a \in A. \end{array} \right.
 \end{aligned}$$

Now, for each commodity $k \in K$, one can substitute for the flow variables the *proportion* of the demand $d(k)$ assigned to arc a , and replace the node demand b_i^k by the unit demand $e_i^k = \text{sgn}(b_i^k)$. Slightly abusing notation, we still denote the flow proportions by x_a^k . Since there exists an optimal extremal solution for the lower program (and the bilevel program as well) one may assume, without loss of generality, that the variables x_a^k are binary-valued, i.e., each commodity flow is assigned to a single path.

Next, we introduce unit commodity toll revenues t_a^k and replace the bilinear term $t_a x_a^k$ by the commodity toll t_a^k , which we force to take the common value t_a whenever the associated flow x_a^k assumes the value ‘one’. These operations yield a mixed-integer program that involves relatively few integer variables, i.e., one per toll arc and per commodity.

$$\begin{array}{l}
 \text{MIP I:} \quad \max_{t, x, \lambda} \sum_{k \in K} \sum_{a \in A_1} n_k t_a^k \\
 \text{subject to} \quad t_a \leq t_a^{\max} \quad \forall a \in A_1 \\
 \\
 \forall k \in K \left\{ \begin{array}{l}
 \sum_{a \in i^-} x_a^k - \sum_{a \in i^+} x_a^k = e_i^k \quad \forall i \in N \\
 \lambda_j^k - \lambda_i^k \leq c_a + t_a \quad \forall a = (i, j) \in A_1 \\
 \lambda_j^k - \lambda_i^k \leq c_a \quad \forall a \in A_2 \\
 \sum_{a \in A_1} (c_a x_a^k + t_a^k) + \sum_{a \in A_2} c_a x_a^k = \lambda_{o(k)}^k - \lambda_{d(k)}^k \\
 -M_k x_a^k \leq t_a^k \leq M_k x_a^k \quad \forall a \in A_1 \\
 -M(1 - x_a^k) \leq t_a^k - t_a \leq M(1 - x_a^k) \quad \forall a \in A_1 \\
 x_a^k \in \{0, 1\} \quad \forall a \in A_1 \\
 x_a^k \geq 0 \quad \forall a \in A_2.
 \end{array} \right.
 \end{array}$$

Note that, in the formulation MIP I, the parameter M_k can be set, for every commodity index k , to any value that exceeds the difference between the cost C_k^∞ of a shortest path that uses only arcs in A_2 and the cost C_k^0 of a shortest path with all tolls set at zero or, if t_a^{\max} is bounded, to t_a^{\max} , simply. As for M , it can assume any value larger than the maximum of the M_k 's. These assignments ensure that formulation MIP I is equivalent to the original bilevel program. In the case where tolls cannot assume negative values, i.e., subsidies are forbidden, these bounds have been refined by Dewez [4].

We now provide two *path-based* formulations for TOP. To this aim, we introduce the set P_k of paths from $o(k)$ to $d(k)$ and denote by I_a^k the set of

elements of P_k that contain a , i.e.,

$$I_a^k = \{p \in P_k \mid a \in P\}, \quad \forall a \in A, \quad \forall k \in K.$$

With each path $p \in P_k$, we associate the indicator variable z_p , which takes the value 1 if path p is used by commodity k , and takes the value 0 otherwise. From the identity

$$x_a^k = \sum_{p \in I_a^k} z_p, \quad \forall a \in A, \quad \forall k \in K,$$

there comes the *arc-path formulation*

$$\text{MIP II:} \quad \max_{t, z, \lambda, s} \sum_{k \in K} \sum_{a \in A_1} n_k t_a^k$$

subject to

$$\forall k \in K \left\{ \begin{array}{ll} \lambda_i^k - \lambda_j^k = c_a + t_a - s_a^k & \forall a = (i, j) \in A_1 \\ \lambda_i^k - \lambda_j^k = c_a - s_a^k & \forall a = (i, j) \in A_2 \\ 0 \leq s_a^k \leq M(1 - \sum_{p \in I_a^k} z_p) & \forall a \in A \\ -M_k \sum_{p \in I_a^k} z_p \leq t_a^k \leq M_k \sum_{p \in I_a^k} z_p & \forall a \in A_1 \\ t_a^k \leq t_a \leq t_a^{\max} & \forall a \in A_1 \\ \sum_{p \in P_k} z_p = 1 & \\ z_p \in \{0, 1\} & \forall p \in P_k, \end{array} \right.$$

where s is the vector of slack variables associated with the dual constraints. The first three constraints ensure that the selected paths are optimal with respect to the current toll vector. The fourth and fifth ones, together with the max operator, ensure that the commodity revenue t_a^k is equal to the true revenue t_a whenever arc a lies on the path actually used by commodity k , hence that the model is consistent. The set of values that can be assumed by the constants M and M_k is the same as that for MIP I.

This formulation involves $\sum_{k \in K} |P_k|$ binary variables. Although this number grows exponentially with the size of the network, it may very well be less than the number of variables involved in MIP I, whenever the number of ‘reasonable’ paths is small. In Section 3, we present a procedure that limits the number of paths to be considered, and consequently makes this approach practical for realistic instances of TOP.

The third formulation, MIP III, is entirely path-based. Let us first introduce T^k , the profit raised from commodity k , as well as L^k , the disutility (cost plus delay) associated with the shortest path p actually used by commodity k . Since at most one path is used for every commodity, we obtain

$$T^k = \sum_{a \in p \cap A_1} t_a$$

and

$$\begin{aligned} L^k &= \sum_{a \in p} (c_a + t_a) \\ &= T^k + \sum_{p \in P_k} z_p \sum_{a \in p} c_a. \end{aligned}$$

This leads to the path formulation MIP III, that involves a smaller number of variables than formulation MIP II.

$$\begin{aligned} \text{MIP III:} \quad & \max_{t, z, L} \sum_{k \in K} n_k T^k \\ & \text{subject to} \quad t_a \leq t_a^{\max} \quad \forall a \in A_1 \\ & \left\{ \begin{array}{l} T^k \leq \sum_{a \in p \cap A_1} t_a + M_k(1 - z_p) \quad \forall p \in P_k \\ \sum_{a \in p \cap A_1} t_a + \sum_{a \in p} c_a - M_k^p(1 - z_p) \leq L^k \leq \sum_{a \in p \cap A_1} t_a + \sum_{a \in p} c_a \quad \forall p \in P_k \\ L^k = T^k + \sum_{p \in P_k} z_p \sum_{a \in p} c_a \\ \sum_{p \in P_k} z_p = 1 \\ z_p \in \{0, 1\} \quad \forall p \in P_k. \end{array} \right. \end{aligned}$$

In this formulation, a suitable value for M_k^p is given by:

$$M_k^p = \sum_{a \in p} c_a + \sum_{a \in p} t_a^{\max} - C_k^0.$$

3 A path generation algorithm

In this section we propose an algorithmic framework that relies on the following three observations:

- lower level solutions correspond to some shortest paths for each origin-destination pair;
- for a given lower level extremal solution (collection of shortest paths), one may efficiently recover a set of revenue-maximizing tolls that is compatible with this solution;
- one may expect to extract higher revenues from toll arcs belonging to paths having low rather than large initial delays. (This should come at no surprise.)

The algorithm generates a sequence of extremal lower level solutions, or *multipath* P , which corresponds to vectors of commodity paths, one per commodity, e.g.,

$$P = (p_1, p_2, \dots, p_{|K|}), \text{ with } p_k \in P_k.$$

We denote by C the set of all multipaths:

$$C = \{(p_1, \dots, p_{|K|}) \mid p_k \in P_k; k = 1, \dots, |K|\}.$$

Given a multipath $P \in C$, we define $c(P)$ as the sum of delays on the arcs belonging to at least one of its paths, i.e.,

$$c(P) = \sum_{k=1}^{|K|} n_k \sum_{a \in p_k} c_a.$$

Without loss of generality, we assume that the elements of C are indexed in nondecreasing order of their respective total delays:

$$c(P^1) \leq c(P^2) \leq \dots \leq c(P^{|C|}).$$

The algorithm explores multipaths in increasing order of total delay, and stops as soon as no progress can be achieved. One iteration of the generic algorithmic scheme is composed of the following operations:

1. Generate the i th multipath P^i ;
2. Update upper bound on total revenue;
3. Optimize toll schedule with respect to current multipath;
4. Update lower bound on total revenue;
5. If lower and upper bounds coincide, stop with an optimal solution to TOP.

The efficiency of the procedure rests on the quality of lower and upper bounds, which are crucial in limiting the scope of the enumeration process, and on the design of an efficient algorithmic procedure for generating multipaths. Those are considered in turn.

3.1 Upper bound

Let p_k^∞ (respectively p_k^0) denote the shortest path from $o(k)$ to $d(k)$ in the graph G obtained by setting all tolls to $+\infty$ (respectively 0) and let α_k^∞ (respectively α_k^0) denote the corresponding delay, i.e.,

$$\alpha_k^\infty = \sum_{a \in p_k^\infty} c_a$$

$$\alpha_k^0 = \sum_{a \in p_k^0} c_a.$$

For a given commodity index k in K , an upper bound on the revenue raised from this commodity k is given by the product of the demand n_k and the gap between α_k^∞ and α_k^0 , i.e.,

$$UB(p_k) = n_k(\alpha_k^\infty - \alpha_k^0).$$

This bound can actually be tightened by making it dependent on the delay of the multipath under consideration:

$$UB(P) = \sum_{k \in K} n_k(\alpha_k^\infty - \sum_{a \in p_k} c_a).$$

3.2 Lower bound

If the lower level solution (multipath) is known a priori, all bilinear constraints become linear, and the resulting program is easy. Its solution yields a toll vector t that maximizes revenue while being compatible with the multipath. This operation is tantamount to solving an inverse problem. Loosely speaking, an inverse optimization problem occurs when one wishes to estimate the parameters of a primary optimization problem whose optimal solution is known a priori. In general, the set of such parameters is not unique, and one may therefore optimize a secondary objective over this set. In the context of toll optimization, one seeks tolls that maximize revenue (the secondary objective) while inducing a predetermined lower level solution, i.e., a toll-compatible multipath. The resulting toll schedule provides the best solution compatible with the multipath, and thus a valid lower bound on the problem's optimum value.

Let $LB(P)$ denote the optimal value of the inverse optimization problem associated with the multipath $P = \{p_k\}_{k \in K}$, i.e.,

$$LB(P) = \max_{t \in D} \sum_{k=1}^{|K|} n_k \sum_{a \in p_k \cap A_1} t_a$$

where

$$D = \left\{ t \mid \sum_{a \in p_k} c_a + \sum_{a \in p_k \cap A_1} t_a \leq \sum_{a \in \bar{p}_k} c_a + \sum_{a \in \bar{p}_k \cap A_1} t_a \quad \forall \bar{p}_k \in P_k \quad \forall k \in K \right\}.$$

While the number of constraints that define the set D is exponential, the associated separation problem can be solved in polynomial time. Indeed, a shortest path oracle can be used to check whether a given toll vector t lies in D , and exhibit, whenever $t \notin D$, a violated constraint. It follows from a result of Grötschel et al [6] that the inverse optimization problem is polynomially solvable. Alternatively, Labbé et al [8] have shown how to reduce the inverse problem to a transshipment problem over a suitably defined network, and for which polynomial algorithms are well known.

3.3 Computation of the i th shortest multipath

The computation of multipaths in increasing order of their fixed costs is an essential part of the sequential algorithm. In this subsection, we describe a procedure to compute the i th shortest multipath

$$P^i = (p_1^{i(1)}, p_2^{i(2)}, \dots, p_{|K|}^{i(|K|)}),$$

where $i(k)$ denotes the ranking of the path associated with commodity k . In particular, the shortest multipath is

$$P^1 = (p_1^1, p_2^1, \dots, p_{|K|}^1).$$

i th shortest multipath

Step 0 [Initialization]

Set $j \leftarrow 1$, and $LIST \leftarrow \emptyset$.

Compute p_k^1 and p_k^2 and set $j(k) \leftarrow 1$, $k = 1, \dots, |K|$.

$$P^1 = (p_1^1, p_2^1, \dots, p_{|K|}^1).$$

Step 1 If $j = i$, stop: the i th multipath has been obtained.

Step 2 [updating of $LIST$]

For all $l \in \{1, \dots, |K|\}$, let $P^{j,l} \leftarrow (p_1, \dots, p_{|K|})$, where $p_k = p_k^{j(k)}$

if $k \neq l$, and $p_l = p_l^{j(l)+1}$. Set $LIST \leftarrow LIST \cup P^{j,l}$.

Set $j \leftarrow j + 1$.

Step 3

Remove the least costly multipath from $LIST$ distinct from P^1, \dots, P^{j-1} , and output this element P^j as the j th shortest multipath.

Step 4

Let $j_0 \in \{1, \dots, j-1\}$ and $l_0 \in \{1, \dots, |K|\}$ such that $P^j = P^{j_0, l_0}$.

Compute $p_{l_0}^{j_0(l_0)+2}$ and set $j_0(l_0) \leftarrow j_0(l_0) + 1$. Return to step 1. \square

The above algorithm requires the knowledge of commodity paths, sorted in increasing order of their costs. To this end, we adopt a procedure proposed by Lawler [10], which we describe, for the sake of completeness. Let the arcs of the directed graph be numbered $1, \dots, m$. For a given path p , let $x_j = 1$ if arc j is contained in p , and $x_j = 0$ otherwise. Given an integer i , the procedure generates the first i shortest paths in sequence.

*i*th-shortest path algorithm

Step 0 [Initialization]

Compute a shortest path $x^{(1)} = (x_1^{(1)}, x_2^{(1)}, \dots, x_m^{(1)})$, without fixing the values of any variables.

$LIST \leftarrow \{x^{(1)}\}$ and $j \leftarrow 1$.

Step 1 [Output the j th shortest path]

Remove the least costly solution from $LIST$ and output this solution, denoted by $x^{(j)} = (x_1^{(j)}, x_2^{(j)}, \dots, x_m^{(j)})$, as the j th shortest path.

Step 2 If $j = i$, stop; the i th shortest path has been obtained.

Step 3 [Update of $LIST$]

Assume that the j th shortest path was obtained by fixing the following conditions

$$\begin{aligned} x_1 = x_2 = \dots = x_q = 1, \\ x_{q+1} = x_{q+2} = \dots = x_s = 0, \end{aligned}$$

where a reordering has been assumed for notational purposes. Leaving these variables fixed as they are, create $m - s$ new shortest path problems that must satisfy the additional conditions

$$\begin{aligned} x_{s+1} &= 1 - x_{s+1}^{(j)}, \\ x_{s+1} &= x_{s+1}^{(j)}, x_{s+2} = 1 - x_{s+2}^{(j)}, \\ &\vdots \\ x_{s+1} &= x_{s+1}^{(j)}, x_{s+2} = x_{s+2}^{(j)}, \dots, x_{m-1} = x_{m-1}^{(j)}, x_m = 1 - x_m^{(j)}. \end{aligned}$$

Compute optimal solutions (i.e, the shortest path subject to conditions above) to each of these $m - s$ problems and place each of the $m - s$ solutions in $LIST$, together with a record of the variables which were fixed for each of them. Set $j = j + 1$. Return to Step 1. □

Remark that the first time Step 3 is executed, $q = s = 0$

3.4 Algorithm specification

We now formally state the algorithm. Let LB^* be the current best profit, P^* the associated multipath, and UB^* the current upper bound. Note that, since the upper bound is non increasing, UB^* is actually the same as $UB(P)$ evaluated at the current multipath. Let N denote the number of distinct multipaths.

Multipath Algorithm

Step 0 [Initialization]

$$LB^* \leftarrow -\infty.$$

$$i \leftarrow 1.$$

Step 1 [Multipath generation and evaluation]

Generate the i th smallest element of C ,

$$P^i = (p_1^{i(1)}, p_2^{i(2)}, \dots, p_{|K|}^{i(|K|)}).$$

$$UB^* \leftarrow \sum_{k \in K} n_k (\alpha_k^\infty - \sum_{a \in P_k^{i(k)}} c_a).$$

Compute $LB(P^i)$ by inverse optimization.

$$\text{Set } LB^* \leftarrow \max\{LB^*, LB(P^i)\}.$$

Step 2 [Stopping criterion]

If $UB^* \leq LB^*$ or $i = N$, stop. The optimal solution is the multipath P^* that has achieved the best lower bound.

$$i \leftarrow i + 1 \text{ and return to step 1.} \quad \square$$

In order to prove the correctness of the algorithm, it suffices to remark that the upper bound UB^* does not increase at each iteration, so that an optimal multipath cannot be missed. Note that the algorithm may have to scan the entire list of multipaths, and that it may terminate with the local upper bound UB^* being *strictly* less than the lower bound LB^* . This can be observed on the single-commodity example illustrated in Figure 1, taken from Labbé et al. [9]. In this example, the first multipath P^1 (a single path in this case) generated is $\{(1 - 2 - 3 - 4 - 5)\}$. Its upper bound is $22 - 6 = 16$ while its lower bound (this easy to check), is equal to 15. The algorithm stops after generating $P^2 = \{(1 - 2 - 4 - 5)\}$, whose upper bound 11 is less than the lower bound of the first path. There does not exist a path that achieves an upper bound equal to the optimal value.

3.5 Redundant paths

A serious drawback of formulations MIP II and MIP III is that all paths between all origin-destination pairs must be enumerated a priori. Obviously,

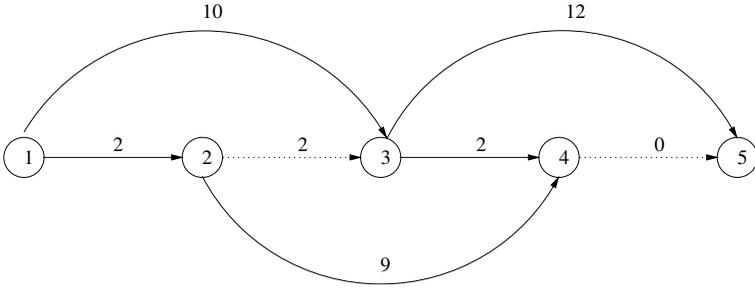


Fig. 1. The multipath algorithm terminates with $LB^* \neq UB^*$

many paths are suboptimal and irrelevant. For instance, one need not consider paths that contain toll-free subpath that are *not* shortest subpaths. Along this line of reasoning, Kraaij [7] constructed a Shortest Path Graph Model (SPGM), equivalent to TOP, where subpaths between toll arcs are replaced by single arcs with cost set to that of a shortest subpath; similarly, subpaths from the origins to the tail of toll arcs, and from the head of toll arcs to the destinations, are shrunk to single arcs. While this preprocessing does not affect the combinatorial nature of the problem, it may reduce computing times by a (roughly) constant factor. Note that, in some cases, the SPGM may contain *more* arcs than the original network, and the computational burden of setting up an SPGM may actually greatly exceed that of solving the resulting problem (see [2]).

While we have not adopted the SPGM formulation, we have implemented a technique for eliminating a subset of dominated paths, according to the criterion outlined in the following lemma. This allows to limit the number of problems created at step 3 of the i th-shortest path algorithm.

Lemma 3.1 (Bouhtou et al [2]) *Consider an instance of TOP where tolls are restricted to be nonnegative. Let p and p_* be two paths between an origin-destination pair $k \in K$, and let $p = (p^1, p^2)$ and $p_* = (p_*^1, p_*^2)$ denote their partition into toll and toll-free arcs, respectively. Assume that $p_*^1 \subseteq p^1$ and that*

$$\sum_{a \in p_*} c_a \leq \sum_{a \in p} c_a.$$

If p , together with a toll schedule t , is optimal for TOP, it follows that the couple (p_, t) is also optimal for TOP.*

Based on the above result, one may reduce the number of problems generated at step 3 of the i th shortest path algorithm by only considering the toll arcs and undominated paths. Suppose, without loss of generality, that the arcs of A_1 are numbered $1, \dots, |A_1|$ and that the j th shortest path contains arcs

$1, \dots, r, r \leq |A_1|$. Suppose, moreover, that the j th shortest path is the shortest path obtained by fixing the following variables

$$\begin{aligned} x_1 &= x_2 = \dots = x_p = 1, \\ x_{r+1} &= x_{r+2} = \dots = x_q = 0, \end{aligned}$$

and the other toll variables (non fixed) verify

$$\begin{aligned} x_{p+1}^{(j)} &= x_{p+2}^{(j)} = \dots = x_r^{(j)} = 1, \\ x_{q+1}^{(j)} &= x_{q+2}^{(j)} = \dots = x_{|A_1|}^{(j)} = 0. \end{aligned}$$

Leaving the fixed variables as they are, create $r-p$ new shortest path problems that must satisfy the additional conditions

$$\begin{aligned} x_{p+1} &= 0, \\ x_{p+1} &= 1, x_{p+2} = 0, \\ &\vdots \\ x_{p+1} &= x_{p+2} = \dots = x_{r-1} = 1, x_r = 0. \end{aligned}$$

4 A block sequential heuristic (BLOSH)

While, as we shall see in the next section, the three MIP reformulations and the exact multipath algorithm allow to tackle medium size problems, the NP-hard nature of TOP will ultimately limit the size of problems that can be solved to prove optimality. The main limitation is due both to the large number of commodities and their interactions. To circumvent the problem, we have implemented a windowing technique that consists in optimizing over a subset of commodities at a time, keeping fixed the paths associated with the (temporarily) fixed commodities. This results in a block sequential heuristic (BLOSH), reminiscent of the Gauss-Seidel approach, well known in optimization. At each iteration, the subproblems are solved using either one of the algorithms presented previously. For our purpose, we have used the MIP III formulation, which involves a small number of binary variables, and proved efficient for solving problems involving a small number of commodities. MIP III was solved using the commercial software CPLEX [3].

More precisely, let us consider a partition K into the set of *active* and *inactive* commodities (origin-destination pairs), i.e., $K = K^1 \cup K^2$. The restricted MIP III formulation then takes the form:

$$\text{MIP III-R: } \max_{t,z,L} \sum_{k \in K} n_k T^k$$

subject to

$$\forall k \in K^1 \left\{ \begin{array}{l} T^k \leq \sum_{a \in p \cap A_1} t_a + M_k(1 - z_p) \quad \forall p \in P_k \\ \sum_{a \in p \cap A_1} t_a + \sum_{a \in p} c_a - M_k^p(1 - z_p) \leq L^k \leq \sum_{a \in p \cap A_1} t_a + \sum_{a \in p} c_a \quad \forall p \in P_k \\ L^k = T^k + \sum_{p \in P_k} z_p \sum_{a \in p} c_a \\ \sum_{p \in P_k} z_p = 1 \\ z_p \in \{0, 1\} \quad \forall p \in P_k \end{array} \right.$$

$$\forall k \in K^2 \left\{ \begin{array}{l} T^k = \sum_{a \in p_k^{i(k)} \cap A_1} t_a \\ L^k \leq \sum_{a \in p \cap A_1} t_a + \sum_{a \in p} c_a \quad \forall p \in P_k \\ L^k = T^k + \sum_{a \in p_k^{i(k)}} c_a. \end{array} \right.$$

In practice, the number of active commodities, $|K^1|$ is small with respect to the number of inactive commodities $|K^2|$. In that case, MIP III-R involves a small number of binary variables and is efficient. A pseudocode for the algorithm is outlined below.

Algorithm BLOSH**Step 0** [initialization]

Compute a feasible paths solution and the associated optimal tax vector (e.g. based on the shortest multipath).

Let \bar{k} be the number of active commodities.

$i \leftarrow 1$.

Step 1

$K^1 \leftarrow \{[(i-1) \bmod |K|] + 1, [(i) \bmod |K|] + 1, \dots, [(i + \bar{k} - 2) \bmod |K|] + 1\}$

$K^2 \leftarrow K \setminus K^1$

Solve MIP III-R and let V^i be its optimal value.

If $V^i = V^{i+1}$, stop.

Step2

$i \leftarrow i + 1$.

Return to step 1.

5 Numerical results

The numerical tests have been performed on randomly generated networks, and give a good idea of the various algorithms' behaviour with respect to the size of the instances. They have been conducted in two steps. We first tested the multipath algorithm on medium-size instances, rapidly showing the limitations of this approach. We then assessed the efficiency of the three MIP formulations, using the commercial MIP solver CPLEX 6.0, versus that of the sequential heuristic. The tests were performed on a SUN ULTRA60 workstation.

The main parameters of the test problems were $|N|$ (number of nodes), $|A|$ (number of arcs), $|A_1|$ (number of toll arcs), $|K|$ (number of commodities). For each toll-free arc (respectively toll arc), an integer fixed cost c_a was uniformly chosen in the interval $[2,20]$ (respectively $[0,6]$). The origin-destination pairs were uniformly chosen, and their demands set to uniform random variables on the interval $[20,100]$. Finally, to ensure connectivity of the underlying graph and the existence of toll-free paths for each origin-destination pair, a Hamiltonian circuit composed only of toll-free arcs was integrated within the network.

Tables 1,2 and 3 illustrate the (in)efficiency of the multipath algorithm. Each table presents the solution of 10 randomly generated problems. We observe that the multipath approach can solve small to medium scale problems, but fails on larger instances. Whenever the number of commodities increases, the approach rapidly shows its limits, mainly due to the large number of multipaths with similar length values. This disappointing performance is due to the existence of nearly identical multipaths. This resulted in upper bounds

that decrease very slowly, as well as lower bounds that increase by steps, after having stalled for several iterations. This can be observed on Figures 2, 3 and 4.

The next tables illustrate the performance of CPLEX on the three MIP formulations, comparing with algorithm BLOSH, for various problem sizes. Each MIP problem was solved with the default parameters of CPLEX 6.0. Running times include the elimination of dominated multipaths.

Algorithm BLOSH was initiated with a shortest multipath. The first \bar{k} active commodities (set to 20) were chosen as follows: we solved independent TOP problems, one for each commodity and reordered them from higher to lower revenue. These revenues were obtained by inverse optimization.

Random networks were generated for various values of the main parameters ($|N|$, $|A|$, $|A_1|$ and $|K|$). The results of our computational experiments are presented in Tables 4 to 9, where each line corresponds to one specific instance. Column headers show the instance number, as well as the running times and the number of Branch-and-Bound nodes explored by CPLEX, for the three MIP formulations. We also indicated, in the BLOSH column, the percentage of optimality reached by the algorithm, 100% indicating that an optimal solution was obtained.

Tables 4 and 5 provide the running times for the four algorithms. While no clear conclusion could come out concerning the average number of nodes, MIP III came out the winner, as it could process each node much faster. On these medium-sized problems, Algorithm BLOSH converged to an optimal solution on all but two instances, with running times slightly less, on the average, than those of MIP III. In Tables 6, 7, 8 and 9, we focused on the following issues:

- How close is the solution provided by BLOSH to an optimal solution?
- How many iterations are required for BLOSH to converge?

On the larger instances, it was not always possible to answer the first question, due to excessive running times. Indeed, MIP III could not reach an optimal solution, or *prove* that such solution was reached, within the imposed time limit set respectively at 10 000 seconds (Tables 6 and 7) and 15 000 seconds (Table 9). Three instances (marked with an asterisk in Table 7) were subsequently allowed 40 000 of CPU time and yet failed to reach an optimum. Actually, the best solution achieved by MIP III was improved for most instances reported in Tables 8 and 9, i.e., whenever the deviation from MIP III's best value exceeded 100 in the corresponding entry of the percentage column.

Finally, note that it is not straightforward to compare our numerical results with those obtained by the MIP formulation of Bouhtou et al. [2]. Indeed:

- The nature of the problems generated in their paper is quite different from ours. First, the number of paths between OD pairs is less than 3, on average it is of the order of 30 undominated paths for our instances.
- The proportion of toll arcs is much higher in our experiments.

- Computers used are from different generations.

This being said, the ratio of improvement between MIP I and MIP III is comparable to the ratio observed in [2] between MIP I (AMIP according to their notation) and their path-based formulation PMIP, once the computational time associated with the generation of the SPGM has been taken into account. Note that, on the set of problems considered by these authors, most of the running time is spent in the preprocessing phase.

6 Conclusion

In this paper, we have proposed new approaches, based on path variables, for addressing an NP-hard problem having applications in the context of optimal pricing. The two main results of the paper were to assess the quality of MIP reformulations of TOP, and to show that the best formulation (MIP III) could be used as the core of a promising heuristic procedure (BLOSH). We are currently working along two lines of attack. First, we wish to embed the most efficient procedures within a decomposition framework. Second, sophisticated techniques (partial inverse optimization) are being investigated, with the aim of improving the upper bound of the multipath algorithm (which is typically of very bad initial quality) and of reducing the number of multipaths explored in the course of the algorithm.

References

1. Brotcorne, L, Labbé, M., Marcotte, P., Savard, G., “A bilevel model for toll optimization on a multicommodity transportation network”, *Transportation Science*, 35, 345–358, 2001.
2. Bouhtou, M., van Hoesel, S., van der Kraaij, A., Lutton, J.-L., “Tariff optimization in networks”, Research Memorandum 041, METEOR, Maastricht Research School of Economics of Technology and Organization, 2003.
3. CPLEX, ILOG CPLEX, v6.0, 2000.
4. Dewez, S., *On the toll setting problem*. PhD thesis, Université Libre de Bruxelles, Institut de Statistique et de Recherche Opérationnelle, 2004.
5. Grigoriev, A., van Hoesel, S., van der Kraaij, A., Uetz M., Bouhtou, M., “Pricing Network Edges to Cross a River”, Research Memorandum 009, METEOR, Maastricht Research School of Economics of Technology and Organization, 2004.
6. Grötschel, M., Lovász, L., Schrijver, A., “The ellipsoid method and its consequences in combinatorial optimization”, *Combinatorica*, 1, 169–197, 1981.
7. van der Kraaij, A., *Pricing in networks*. PhD thesis, Proefschrift Universiteit Maastricht, 2004.
8. Labbé, M., Marcotte, P., Savard, G., “A bilevel model of taxation and its applications to optimal highway pricing”, *Management Science*, 44, 1608–1622, 1998.

9. Labbé, M., Marcotte, P., Savard, G., “On a class of bilevel programs”, In: Non-linear Optimization and Related Topics, Di Pillo and Giannessi eds., Kluwer Academic Publishers, 183-206, 1999.
10. Lawler, E.L., “A procedure to compute the K best solutions to discrete optimization problems and its application to the shortest path problem”, *Management Science*, 18, 401-405, 1972.
11. Marcotte, P., Savard, G. and Semet, F. “A bilevel programming approach to the travelling salesman problem”, *Operations Research Letters*, 32, 240-248, 2004.
12. Roch, S., Savard, G., Marcotte, P., “Design and analysis of an algorithm for Stackelberg network pricing”, *Networks*, 46, 57-67, 2005.

Instances	Nodes	time (s)	gap (%)
1	42	0.11	0.00
2	287	0.12	0.00
3	27	0.13	0.00
4	102	0.17	0.00
5	21	0.14	0.00
6	8	0.12	0.00
7	152	0.17	0.00
8	86	0.16	0.00
9	54	0.17	0.00
10	194	0.20	0.00

Table 1. Problems with 60 nodes, 200 arcs, 20 tolled arcs and 10 O-D-pairs

Instances	Nodes	time (s)	gap (%)
1	1061001	14022.78	18.04
2	1161001	14006.93	7.46
3	2313001	14006.94	8.91
4	495542	2559.90	0.00
5	1375001	14002.36	3.78
6	1270001	14021.93	21.00
7	1220001	14024.66	20.70
8	78019	1017.52	0.00
9	1501001	14012.94	3.92
10	1219001	14015.69	8.86

Table 2. Problems with 60 nodes, 200 arcs, 40 tolled arcs and 20 O-D-pairs

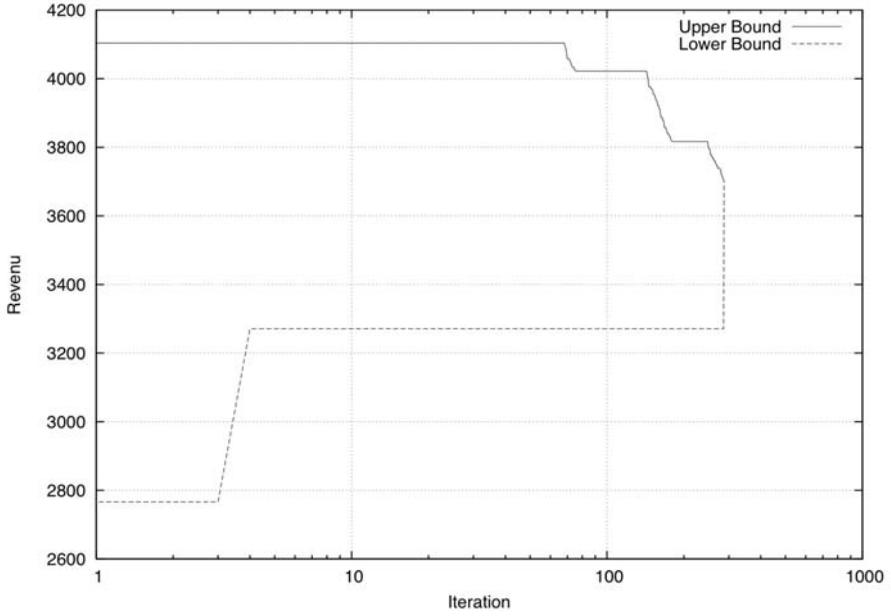


Fig. 2. Upper and lower bounds: instance 1 (60,200,20,10)

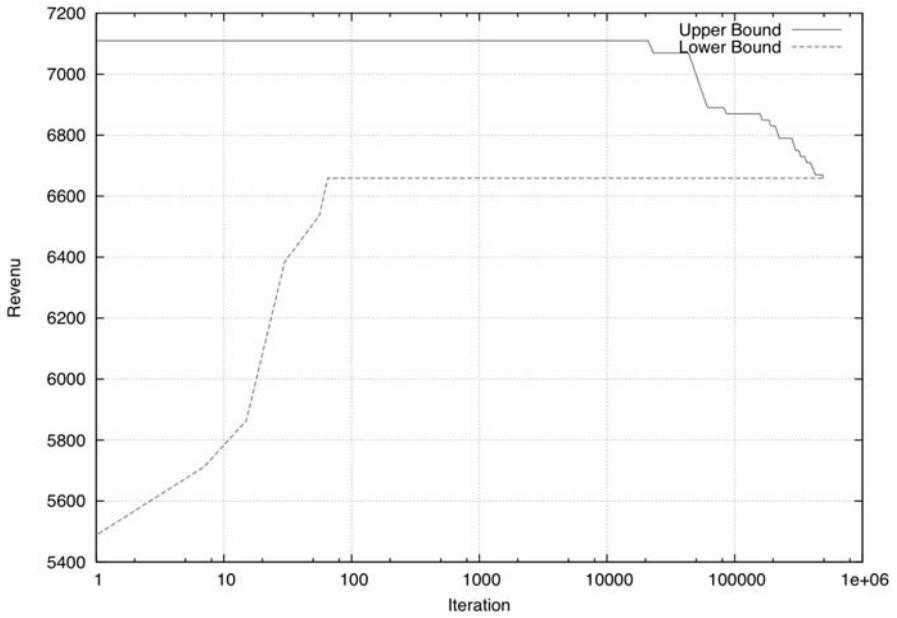


Fig. 3. Upper and lower bounds: instance 1 (60,200,40,20)

Instances	Nodes	time (s)	gap (%)
1	550001	14010.68	39.20
2	933001	14029.33	14.03
3	849001	14019.97	17.17
4	482001	14012.71	28.91
5	729001	14019.96	30.34
6	697001	14012.32	21.81
7	645001	14019.11	28.36
8	645001	14013.17	16.05
9	1016001	14015.23	25.54
10	1112001	14026.46	39.12

Table 3. 90 nodes, 300 arcs, 60 tolled arcs and 40 O-D-pairs

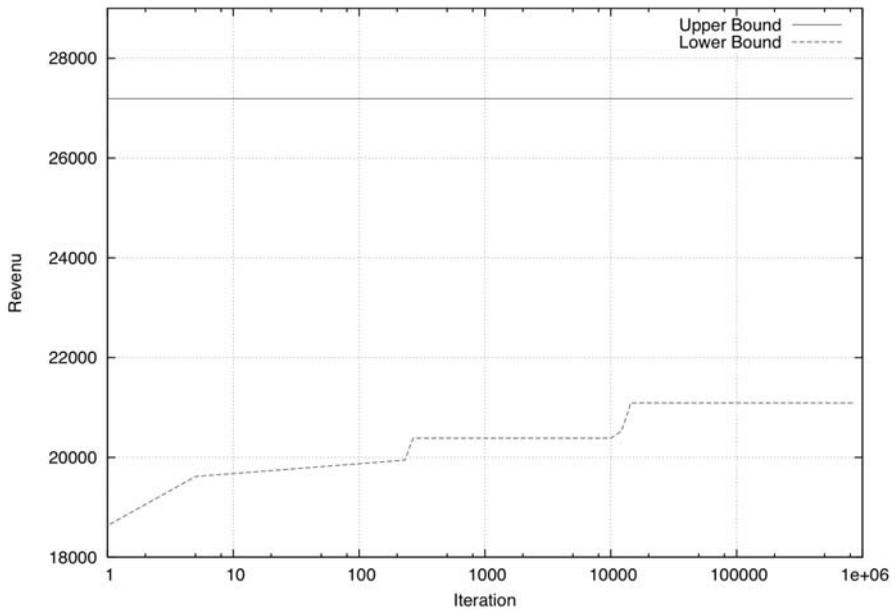


Fig. 4. Upper and lower bounds: instance 1 (60,300,60,40)

Instance	MIP I		MIP II		MIP III		BLOSH	
	Nodes	time (s)	Nodes	time (s)	Nodes	time (s)	%	time (s)
1	48	9.48	44	2.50	46	0.97	100.00	1.32
2	30	3.98	32	2.02	20	0.88	100.00	1.42
3	30	6.70	42	1.63	41	0.47	100.00	0.77
4	144	21.00	45	4.98	67	1.03	100.00	1.59
5	29	8.13	15	1.69	26	0.46	100.00	0.73
6	40	5.97	32	3.31	88	1.02	100.00	1.37
7	36	7.28	55	3.76	62	0.99	100.00	1.39
8	14	5.37	16	1.86	29	0.63	100.00	0.77
9	62	9.70	50	2.91	52	0.59	100.00	0.80
10	12	4.16	9	1.07	16	0.42	100.00	0.58

Table 4. 60 nodes, 20 O-D pairs, 200 arcs, 40 tolled arcs.

Instance	MIP I		MIP II		MIP III		BLOSH	
	Nodes	time (s)	Nodes	time (s)	Nodes	time (s)	%	time (s)
1	5010	5998.84	77500	19806.00	6881	34.11	99.76	12.51
2	496	147.98	232	114.89	1015	3.85	100.00	3.80
3	87	79.23	49	7.63	154	2.46	100.00	3.49
4	243	146.00	361	51.45	479	3.61	100.00	4.44
5	1076	830.20	1618	473.02	622	3.99	100.00	4.36
6	346	808.50	312	259.06	1264	8.61	100.00	7.74
7	425	237.03	1903	260.51	1829	6.87	100.00	5.27
8	145	67.19	295	47.39	856	7.34	100.00	8.04
9	736	2350.88	1129	904.61	3638	19.97	98.29	8.34
10	264	274.21	182	43.40	179	5.00	100.00	6.61

Table 5. 90 nodes, 40 O-D pairs, 300 arcs, 60 tolled arcs.

Instance	MIP III		BLOSH	
	Nodes	time (s)	%	time (s)
1	61897	460.00	97.08	25.37
2	2570196	29661.00	99.21	44.30
3	3542	38.00	98.81	11.12
4	202271	1027.00	97.76	20.52
5	38777	120.00	99.42	10.88
6	43588	239.00	98.20	16.05
7	378033	2609.00	98.75	33.67
8	133873	1002.00	98.39	26.09
9	3170	42.00	99.95	11.65
10	12549	181.00	99.24	31.57

Table 6. 120 nodes, 60 O-D pairs, 400 arcs, 80 tolled arcs.

Instance	MIP III		BLOSH	
	Nodes	time (s)	%	time (s)
1	800166	11304	97.15	122
2	852734	5739	97.46	66
3	3615113	39555	99.28	102
4	3059449	27559	100	91
5(*)	518756	10056	96.22	162
6	2129948	35160	99.39	127
7	182957	2868	100	133
8(*)	432250	10057	99.86	170
9(*)	1905100	10045	102.06	132
10	488533	5116	97.54	95

Table 7. 150 nodes, 80 O-D pairs, 600 arcs, 120 tolled arcs.

Instance	MIP III		BLOSH	
	Nodes	time (s)	%	time (s)
1	532884	10080	112.74	108
2	610662	10086	99.65	118
3	361924	10121	121.45	182
4	580082	10091	131.76	126
5	711571	10076	98.53	103
6	626277	10078	104.40	103
7	857079	10068	100.39	84
8	430176	10102	103.76	145
9	342673	10107	103.94	158
10	426247	10094	107.84	138

Table 8. 200 nodes, 100 O-D pairs, 800 arcs, 160 tolled arcs.

Instance	MIP III		BLOSH	
	Nodes	time (s)	%	time (s)
1	11591	15998	133.66	4991
2	20794	15181	123.98	4430
3	14927	15753	132.47	5139
4	47401	13597	114.17	2880
5	16987	14649	119.20	3962
6	36033	13690	117.84	2992
7	37476	13823	116.80	3006
8	24560	14778	109.41	4040
9	24274	15144	122.73	4450
10	16280	15195	118.39	4425

Table 9. 500 nodes, 200 O-D pairs, 5000 arcs, 1000 tolled arcs.



<http://www.springer.com/978-0-387-34220-7>

Optimization with Multivalued Mappings
Theory, Applications and Algorithms
Dempe, S.; Raskolnikova, P. (Eds.)
2006, XII, 276 p. 16 illus., Hardcover
ISBN: 978-0-387-34220-7