An exact algorithm for the network pricing problem

L. Brotcorne a,*, F. Cirinei b, P. Marcotte c, G. Savard d

a INRIA Lille Nord Europe, Parc Scientifique de la Haute Borne, 40 Avenue Halley, BAT A Park Plaza 59650 Villeneuve d'Ascq, France
b ExPretio, 200 Laurier Street West, Suite 400, Montréal (QC), H2T 2N8, Canada
c CIRRELT and Département d’Informatique et de Recherche Opérationnelle, Université de Montréal C.P. 6128, Succursale Centre-Ville, Montréal (QC), H3C 3J7, Canada
d GERAD and Département de Mathématiques et de Génie Industriel, École Polytechnique de Montréal C.P. 6079, Succursale Centre-Ville, Montréal (QC), H3C 3A7, Canada

A R T I C L E   I N F O

Article history:
Received 16 December 2008
Received in revised form 5 March 2010
Accepted 4 September 2010
Available online 11 November 2010

Keywords:
Pricing
Bilevel programming
Networks
Column generation
Combinatorial optimization

A B S T R A C T

This work focuses on an improved exact algorithm for addressing an NP-hard network pricing problem. The method involves an efficient and partial generation of candidate solutions, a recursive scheme for generating improved upper bounds, and a column generation procedure for solving the network-structured subproblems. Its efficiency is assessed against both randomly generated instances involving three distinct topologies as well as instances based on real life situations in telecommunication and freight transportation.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Let $G = (\mathcal{N}, \mathcal{A})$ be a network with node set $\mathcal{N}$ and arc set $\mathcal{A}$, where $\mathcal{A}$ is partitioned into two subsets $\mathcal{A}_1$ and $\mathcal{A}_2$, with $\mathcal{A}_1$ denoting a subset of links subject to tariffs set by an operator, and $\mathcal{A}_2$ its complement $\mathcal{A} - \mathcal{A}_1$. Each arc $(i, j) \in \mathcal{A}_1$ (resp. $(i, j) \in \mathcal{A}_2$) is endowed with a unit cost $c_{ij}$ (resp. $d_{ij}$). To each origin–destination couple $(o(k), d(k))$ ($k \in \mathcal{K}$) we associate a commodity $k$, a demand $n_k$ and the nodal demand vector $b^k$ defined as:

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

Following Labbé et al. [1], the network pricing problem (NPP in short) consists in maximizing the revenue raised from tariffs $T^\alpha$ set on arcs $\alpha \in \mathcal{A}_1$, under the assumption that commodity flows are assigned to shortest paths with respect to the generalized cost, i.e., the sum of the original cost and tariffs. It can be mathematically formulated as the bilevel program

$$\text{NPP} : \max_{T, x, y} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_1} T^\alpha_{ij} x^k_{ij}$$

* Corresponding author. Tel.: +33 3 59 57 78 82; fax: +33 3 59 57 78 50.
E-mail addresses: Luce.Brotcorne@inria.fr (L. Brotcorne), Fabien.Cirinei@expretio.com (F. Cirinei), Marcotte@iro.umontreal.ca (P. Marcotte), Gilles.Savard@polytechnique.ca (G. Savard).

1572-5286/$ – see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.disopt.2010.09.003
In order that the problem be well posed, we make the standard assumption that there exists at least one tariff-free path for each origin–destination pair. This rules out the trivial solution that could be achieved by setting the components of $T$ to arbitrarily large values.

Two comments are in order with respect to the above formulation. First, note that tariffs are not restricted to be nonnegative. From a mathematical point of view, this leads to a challenging situation where tariffs cannot be bounded a priori, and where complex interactions between taxes and subsidies occur. Next, the model implicitly assumes that, for a given tariff vector, whenever there exist several cheapest paths for some origin–destination pair, then flows are assigned to the cheapest path that maximizes the revenue of the leader (1). Such an assumption is justified by the fact that ties could be broken through the introduction of an arbitrarily small perturbation of the optimal tariff vector. Alternatively, one could adopt the random utility framework, where path assignment is performed according to some discrete choice model (logit, probit, or variants thereof), but such an extension would call for an entirely different algorithmic approach, and lies beyond the scope of the present paper.

The Network Pricing Problem is closely related to product line pricing, a classical and difficult problem in economics (see [2] for instance). As is the case for the latter, the NPP has been proved to be NP-hard by [1], a result later refined by Roch et al. [3] and Grigoriev et al. [4]. While heuristic algorithms have been developed for the NPP [5,6] or variants thereof have been studied [7–9], the present paper is concerned with exact approaches for NPP without network or tariff restrictions. One such approach is based on replacing the lower level shortest path problems by its optimality conditions, yielding a single level problem that can be easily reformulated as a mixed integer program and solved by a commercial software. Alternatively, as proposed also by Didi Biha et al. [10], one can view the NPP as a pure combinatorial problem whose decision variables are origin–destination paths. For a given set of paths, one looks for a tax vector $T$ that maximizes the revenue $Tx$, while simultaneously being compatible with the designated path set being ‘shortest’ with respect to the cost vector $c + T$.

In this paper, we follow the latter approach and design an algorithm whose innovative features include:

- an efficient path generation procedure;
- the use of column generation for the determination of optimal tariffs compatible with a given path set;
- the computation of ‘tight’ upper bounds for the revenue;
- the design of an efficient diversification phase.

Throughout the algorithm, we exploit the network structure of the problem for maximum efficiency. While all four components are key to a successful implementation, we stress the difficulty of devising tight upper bounds for the NPP, especially when tariffs are unrestricted in sign. The paper is organized as follows. Based on a mixed integer reformulation of NPP, we introduce our algorithmic framework (Section 2). The next three sections are devoted to a detailed description of the main components: path evaluation (Section 3), path generation (Section 4), and upper bound computation (Section 5). This leads to a formal description of the algorithm (Section 6) and to numerical results performed on randomly generated instances corresponding to three network topologies, as well as instances based on real life instances in freight transportation and telecommunication (Section 7).

2. Basic algorithmic framework

In this section we introduce an algorithmic framework constructed around the path sets used by the follower. The following notion will be used throughout the paper.

**Definition.** A multipath $P$ is defined as a set of feasible paths, one per each origin–destination pair.

It is natural to expect that high revenues will be generated by short paths so that, subsequently, multipaths will be sorted by their initial cost, each path being weighted by the demand of the associated origin–destination pair. For a given multipath $P$, let $U_0$ denote the objective value of the lower level, i.e. the sum (weighted by origin–destination demands) of the shortest path costs corresponding to null tariffs. Also let $U_\infty$ denote the corresponding total cost when tariffs are set to infinite values, i.e., flows are assigned to the tariff free paths. It is intuitive and actually correct that the quantity $U_\infty - U_0$ yields an upper bound on the maximal revenue that can be achieved under the restriction that the multipath $P$ is an assignment of flows to shortest paths with respect to the total cost $c + T$. In practice, it has been observed that the quality of this bound is poor, and one of the contributions of this paper is the computation, at increased computational cost, of a much improved upper bound.
Fig. 1. A two-commodity network with its fixed cost component.

Actually, the optimal revenue compatible with $P$, thus the best upper bound possible for $P$, can be obtained by solving an inverse optimization problem, whereby one computes revenue-maximizing tariffs that are compatible with a given multipath being composed of shortest paths with respect to the total cost structure. This yields a feasible solution to the network pricing problem, and thus a global lower bound $L(P)$ on the network pricing problem. Note that this bound is set to minus infinity if no tariff vector is compatible with $P$, i.e., no tariff vector can induce flows to be assigned to travel on the paths of $P$. Such instances can trivially be constructed when the number of origin–destination pairs is at least two.

Now, if multipaths are sorted and evaluated in decreasing order of their potential revenues (upper bounds), the algorithm can be halted as soon as the best revenue achieved so far exceeds some upper bound on the potential revenue of the multipath under investigation. Since performing an inverse optimization at each step of the solution procedure is costly, the improved upper bound is designed to strike the right balance between early termination, which requires a good upper bound, and the actual time required to actually compute the upper bound.

In the generic iteration of the procedure outlined below, which matches that of Didi Biha et al. [10], $P^*$ denotes the multipath corresponding to the largest revenue achieved up to iteration $i - 1$, and $T^*$ the corresponding tariff vector. One may assume a lower bound of zero for the revenue, which corresponds for instance to setting $T$ to zero (or to plus infinity, for that matter).

A generic iteration

**Step 1. Multipath generation**

- Let $P^i$ be the multipath having the $i$th largest potential revenue $U(P^i)$.

**Step 2. Multipath evaluation (lower bound)**

- Optimize tariffs with respect to $P^i$ to obtain the lower bound $L(P^i)$ and the tariff vector $T^i$.
- If $L(P^i) > L^*$, then set $L^*$ to $L(P^i)$, $P^*$ to $P^i$, and $T^*$ to $T^i$.

**Step 3. Optimality test**

- If $L^* \geq U(P^i)$ then the optimal multipath is $P^*$ and the optimal tariff vector is $T^*$. Stop.
- $i \leftarrow i + 1$ and return to Step 1.

To illustrate the algorithm, let us consider the two-commodity example depicted in Fig. 1, where dotted arcs $(5, 2), (5, 6), (3, 6)$ are subject to tariffs, and demand for transportation is set to 1 from 1 to 2, and 1 from 3 to 4. In Table 1, all 9 multipaths are listed, together with their respective upper and lower bounds.

While the optimal solution corresponds to the multipath $P^1$ having the largest upper bound (14), this need not be always the case. Actually, an alternative optimal solution is achieved at $P^2$, whose upper bound is inferior (12) to that of $P^1$, and one has to wait until the second iteration before confirming that $P^1$ is indeed optimal.

The following sections deal with the main features of the algorithm.

### 3. Multipath evaluation (inverse optimization)

This section is devoted to the recovery of a set of revenue-maximizing tariffs compatible with a given multipath. This operation yields a feasible solution to the NPP, and hence a valid lower bound on its objective. To this aim, we replace the lower level problem of NPP by its primal–dual optimality conditions, to obtain the bilinear programming formulation

\[
\text{BILIN : } \max_{T, x, y, \lambda} \sum_{k \in K} \sum_{(i,j) \in A_1} T_{ij} x_{ij}^k
\]

s.t. for all $k$, \(\sum_{(i,j) \in A_1} (x_{ij}^k + y_{ij}^k) - \sum_{(j,i) \in A_2} (x_{ji}^k + y_{ji}^k) = b_k^i, \forall i \in N,\)

\[
x_{ij}^k \geq 0, \forall (i,j) \in A_1,
\]

\[
y_{ij}^k \geq 0, \forall (i,j) \in A_2.
\]
whose objective can be expressed, using

Now, for a given multipath, the flow vectors \(x\) and \(y\) are fixed, and BILIN reduces to the (inverse) linear program

\[
\text{IO} : \quad \max_{\tau,k} \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_1} n^k T_{ij}
\]

\[
\text{s.t. for all } k \quad \lambda_j^k - \lambda_i^k \leq c_{ij} + T_{ij}, \quad \forall (i,j) \in \mathcal{A}_1 | x_{ij}^k = 0,
\]

\[
\lambda_j^k - \lambda_i^k \leq d_{ij}, \quad \forall (i,j) \in \mathcal{A}_2 | y_{ij}^k = 0.
\]

\[
(\lambda_j^k - \lambda_i^k) x_{ij}^k = 0, \quad \forall (i,j) \in \mathcal{A}_1,
\]

\[
(\lambda_j^k - \lambda_i^k) y_{ij}^k = 0, \quad \forall (i,j) \in \mathcal{A}_2.
\]

whose objective can be expressed, using (17) and (18), as

\[
\sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_1} n^k T_{ij} = \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_1} n^k (\lambda_j^k - \lambda_i^k - c_{ij}),
\]

\[
= \sum_{k \in \mathcal{K}} n^k \left( \sum_{(i,j) \in \mathcal{A}_1} (\lambda_j^k - \lambda_i^k) - \sum_{(i,j) \in \mathcal{A}_1} c_{ij} - \sum_{(i,j) \in \mathcal{A}_2} (\lambda_j^k - \lambda_i^k) \right),
\]

\[
= \sum_{k \in \mathcal{K}} n^k \left( \lambda_j^k - \lambda_i^k - \sum_{(i,j) \in \mathcal{A}_1} c_{ij} - \sum_{(i,j) \in \mathcal{A}_2} d_{ij} \right).
\]

The linear programming dual of IO, whose formulation is

\[
\text{DIO} : \quad \min \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_1} c_{ij} z_{ij}^k + \sum_{k \in \mathcal{K}} \sum_{(i,j) \in \mathcal{A}_2} d_{ij} z_{ij}^k
\]

\[
\text{s.t. for all } k \quad \sum_{k \in \mathcal{K}} z_{ij}^k = 0, \quad \forall (i,j) \in \mathcal{A}_1,
\]

\[
\sum_{(i,j) \in \mathcal{A}_1} z_{ij}^k - \sum_{(i,j) \in \mathcal{A}_2} z_{ij}^k = b_i^k, \quad \forall i \in \mathcal{N},
\]

\[
z_{ij}^k \geq 0, \quad \forall (i,j) \in \mathcal{A}_1 | x_{ij}^k = 0.
\]
strategy for addressing the inverse optimization problem. The DIOS subproblems correspond to easily solvable shortest path problems, which explains our choice of a decomposition.

The additional constraints (19). An optimal tariff vector $T$ can be readily extracted from its dual solution $\lambda$ and, as shown by Labbé et al. [1], an economic interpretation of this solution can be provided. Moreover, in the single-commodity case, the dual problem reduces to a shortest path problem in a modified network.

We propose to solve DIO by Dantzig–Wolfe decomposition, constraints (19) playing the role of complicating constraints. For each commodity $k$, we introduce the set $\Delta^k$ of flows $z^k$ satisfying constraints (20)–(24). Each solution $z^k \in \Delta^k$ corresponds to the sum of a convex combination of extreme points (corresponding to elementary origin–destination paths), and a linear combination of extreme rays (corresponding to elementary cycles):

$$z^k = \sum_{p \in p^k} \omega_p Z_p + \sum_{p \in P} \rho_p \tilde{Z}_p$$

such that

$$\sum_{p \in p^k} Z_p = 1, \quad \sum_{p \in P} Z_p = 1,$$

$$Z_p \geq 0, \quad \forall p \in p^k \setminus p^k^*, \quad Z_p \geq 0, \quad \forall p \in P \setminus \tilde{P},$$

$$Z_p \text{ free,} \quad \forall p \in p^k^*,$$

where $p^k$ (resp. $p^k^*$) denotes the index set of elementary paths (resp. elementary paths carrying no flow) belonging to $\Delta^k$, $\omega_p$ is the weight vector associated with path $p$ while $Z_p$ is the flow value on this path, $\tilde{P}$ is the set of indices of cycles of $\Delta^k$, $\rho_p$ and $\tilde{Z}_p$ are vectors associated with cycle $p$ of $\Delta^k$.

The Dantzig–Wolfe reformulation leads to the following path formulation of DIO:

$$\text{DIODW :} \quad \min \sum_{k \in K} \sum_{p \in p^k} C_p Z_p + \sum_{p \in P} \tilde{C}_p \tilde{Z}_p$$

s.t $- \sum_{k \in K} \sum_{p \in p^k} l_{a,p} Z_p = \sum_{p \in P} l_{a,p} \tilde{Z}_p = 0, \quad \forall a \in A_1,$

$$\sum_{p \in p^k} Z_p = n^k, \quad \forall k \in K,$$

$$Z_p \geq 0, \quad \forall p \in p^k \setminus p^k^*,$$

$$Z_p \text{ free,} \quad \forall p \in p^k^*,$$

$$\tilde{Z}_p \geq 0, \quad \forall p \in \tilde{P},$$

where $C_p$ (resp. $\tilde{C}_p$) is the path cost (resp. cycle cost). In the above, the indicator $l_{a,p}$ takes value 1 if arc $a$ belongs to path or cycle $p$, 0 otherwise.

This master problem is addressed via column generation. Then, paths or cycles with negative reduced costs for DIODW are identified by solving $k$ single-commodity linear subproblems (DIOS).

Let $\lambda^k$ and $T_a$ be the dual variables associated with constraints (26) and (25), respectively. The reduced cost associated with $Z_p, p \in p^k$ is $C_p + \sum_{a \in A_1} l_{a,p} T_a - \lambda^k$, and the reduced cost associated with $\tilde{Z}_p, p \in P$ is $C_p + \sum_{a \in A_1} l_{a,p} T_a$. The solution of the restricted master program is optimal for the unrestricted master DIODW if there hold

$$\min_{p \in p^k} \left( C_p + \sum_{a \in A_1} l_{a,p} T_a \right) \omega_p - \lambda^k \geq 0, \quad \forall k \in K,$$

$$\min_{p \in P} \left( C_p + \sum_{a \in A_1} l_{a,p} T_a \right) \rho_p \geq 0.$$  

Otherwise, variables with negative reduced costs and the corresponding variables enter the Restricted Master Program.

The column generation scheme is initialized with tariff-free shortest paths, together with the fixed path used by the follower associated with commodity $k$, while the cycle set $\tilde{P}$ is initially empty. For each commodity $k$, since the only free flow path (the one where flows are not restricted to be nonnegative) is a member of the initial $P^k$ set, it turns out that the DIOS subproblems correspond to easily solvable shortest path problems, which explains our choice of a decomposition strategy for addressing the inverse optimization problem.
4. Multipath generation

Let us consider the set \( C \) of all multipaths:
\[
C = \{ (p_1, \ldots, p_{|\mathcal{K}|}) | p_k \in \mathcal{P}_k, k = 1 \ldots |\mathcal{K}| \}.
\]

Our algorithm probes the multipaths in decreasing order of their potential revenues. As mentioned previously, the upper bound on the revenue of a multipath can be set to the difference between the costs of shortest paths with respect to infinite and null tariffs. Alternatively, one could use the improved upper bounds that will be introduced in Section 5. In both cases, potential revenue computation will require the (partial) enumeration of paths in increasing order of their length. This list is obtained through the implementation of a variant of Yen’s deviation algorithm for computing the \( K \) shortest paths [11].

Yen’s algorithm is based on pseudotrees containing \( K \) shortest paths between 2 nodes \( o \) and \( d \), determined from deviation nodes. More precisely, a set of candidate paths \( \chi \) is initialized with the shortest path between \( o \) and \( d \). At iteration \( j \) the \( j \)th shortest path \( p^j \) is selected as the shortest one in \( \chi \). Next \( \chi \) is updated by including a subset of deviation paths between \( o \) and \( d \). The first deviation path is obtained by merging the subpath of \( p^j \) from node \( o \) to the associated deviation node \( v_j \), and the path from node \( v_j \) to node \( d \) where the outgoing arc from \( v_j \) in \( p^j \) is forbidden. The deviation node associated with the resulting path is the node next to node \( v_j \). Note that this deviation path may not exist. Alternative deviation paths are also computed from \( p^j \) by considering nodes following \( v_j \) on \( p^j \) as deviation nodes.

In order to speed up the calculations, the algorithm is applied on the ‘Shortest Path Graph Model’ introduced by Bouhtou et al. [8], where paths between tariff arcs are replaced by a single arc endowed with the cost of a shortest tariff-free path.

Now, let us denote the \( i \)th multipath by
\[
p^i = (p^{i(1)}_1, \ldots, p^{i(|\mathcal{K}|)}_{|\mathcal{K}|}),
\]
where \( p^{i(k)}_k \) is the \( i(k) \)th shortest path associated with commodity \( k \). Starting with \( p^0 \), multipaths are generated by increasing the path rank of a single commodity. To prevent redundancy in the enumeration process, we associate with each multipath a label \( l \) (an integer number between 1 and \( |\mathcal{K}| \)) that represents the index of the last commodity whose path rank has been updated. We enforce the rule that a multipath can only be generated from its father by increasing the rank of a commodity path whose index is larger than or equal to the label. A natural ordering of multipaths is obtained by increasing the path rank of the first commodity, then the path rank of the second, and so on. The process is illustrated in Fig. 2, with and without the use of labels.
Fig. 3. Upper bound on the revenue associated with a multipath.

Labels induce a partition of multipaths into:

- the **prefix**: the set of commodities whose indices are strictly lower than label \( l \). Paths associated with the prefix commodities are fixed in the multipath enumeration process since the paths associated with commodities of index \( j < l \) cannot be modified. All descendants of a multipath inherit the same prefix.
- the **label** \( l \): the commodity with label \( l \).
- the **suffix**: the set of commodities whose indices are strictly greater than the label \( l \). The path associated with each suffix commodity is the shortest path corresponding to zero tariffs. This set of paths is not fixed but all multipaths with label \( l \) share the same suffix.

For notational convenience, label \( l \) will denote both the label part of the multipath and the associated commodity associated throughout the remainder of the paper. In the main algorithm, the tree will be explored in a Best-First manner with respect to the improved upper bounds introduced in the next section.

5. Improved upper bounds

Both in the papers by Labbé et al. [1] and Didi Biha et al. [10], it was observed that the upper bound derived from infinite-tariff and tariff-free shortest paths was of poor quality. In this section, we propose tighter bounds which, substituted to \( U(P) \) at Steps 2 and 3 of the algorithmic framework (see Section 2), allow us to prune at an early stage branches of the multipath generation tree illustrated in Fig. 2, and hence to speed up the computation or recognition of an optimal solution. These bounds are based on the partition of a multipath \( P \) into its prefix, label and suffix components, each of these entities being treated separately. The prefix part of the upper bound \( B_{\text{prefix}}(P) \) is the optimal revenue associated with the prefix commodities considered in isolation. Since their paths are fixed, this is simply achieved by solving an inverse linear program, and yields the best possible value, since it coincides with the optimal revenue compatible with the prefix paths. Note that this analysis may show that the prefix is infeasible, i.e., there exists no tariff compatible with the prefix paths, in which case the corresponding branch of the enumeration tree is pruned. The label bound \( B_{\text{label}}(P) \) is simply set to the difference between the cost of the tariff-free shortest path and the cost of the current path with tariffs set at 0, weighted by the demand for the commodity associated with the label.

The nontrivial part of the computation concerns the suffix bound. The computation of the revenue upper bound generated by the suffix commodities \( B_{\text{suffix}}(l) \) is based on the fact that all multipaths with label \( l \) have the same suffix denoted by \( \text{suffix}(l) \). The number of suffixes is thus equal to \(|K| - 1\). The evaluation of the bound \( B_{\text{suffix}}(1), \ldots, B_{\text{suffix}}(|K| - 1) \) is then achieved by recursively applying the exact method (EMA) for a predetermined number of iterations, starting backwards from \( B_{\text{suffix}}(|K| - 1) \). The number of iterations is set to a prespecified value corresponding to the depth in the “tree” representation. Each call requires the computation of the label upper bound and relies on the results of the previous step. Note that, if the number of iterations were not limited, this procedure would determine the optimal solution of the problem associated with its prefix and label components.

From these definitions, we obtain immediately the following proposition stating that a multipath \( \tilde{B}(P) \) upper bound can be set to the minimum value between the sum of the bounds on each term of the partition and the sum of the prefix upper bound and the suffix \((l - 1)\) upper bound (see the upper part of Fig. 3). If this second term is binding, then it is preferable to include commodity \( l \) in the suffix of its predecessor.

**Proposition 1.** Let \( P \) be a feasible multipath with label \( l \). Then

\[
\tilde{B}(P) = B_{\text{prefix}}(P) + \min\{B_{\text{suffix}}(l-1), B_{\text{label}}(P) + B_{\text{suffix}}(l)\}
\]

is a valid upper bound for the multipath \( P \).

The next proposition shows that the improved upper bound related to a node is valid for all its descendants.

**Proposition 2.** Let \( P \) be a multipath with label \( l \) and \( P' \) a descendant from \( P \) with \( l' \geq l \). Then \( \tilde{B}(P) \) is an upper bound on the revenue raised from the multipath \( P' \).
Proof. The proof relies on the multipath decomposition illustrated in Fig. 3.

Case 1. \( \tilde{B}(P_i) = B_{\text{prefix}}(P_i) + B_{\text{suffix}}(l - 1) \).

We first decompose the IO objective value \( v(P'_l) \) associated with \( P'_l \) according to the contribution of each part of the partition into the prefix, the label and the suffix. More precisely we denote by \( v_{\text{prefix}}(P'_l) \) (resp. \( v_{\text{suffix}}(P'_l) \)) the contribution to \( v(P'_l) \) generated by the commodities of the prefix (resp. the commodities of the suffix). \( v(P'_l) = v_{\text{prefix}}(P'_l) + v(l') + v_{\text{suffix}}(P'_l) \). According to the decomposition illustrated in Fig. 3 we have: \( v(P'_l) = v(o'_l) + v(o'_l) + v(o'_l) + v(o'_l) \). As the contribution to \( v(P'_l) \) generated by the commodities of \( o'_l \) is less than or equal to the objective value of the IO solved on the commodities of the set \( o'_l \), which is defined as the revenue upper bound induced by the commodities of \( o'_l \), we have that \( v(o'_l) \leq B(o'_l) \).

By the definition of a descendant, the paths of the commodities in the prefix of \( P_i \) are identical to the paths of set \( o'_l \) in \( P'_l \). Thus \( v(o'_l) \leq B(o'_l) = B_{\text{prefix}}(P_i) \). Moreover by the definition \( B_{\text{suffix}}(l - 1) \) is an upper bound on the revenue generated by the commodities of suffix \((l - 1)\). As the commodities of suffix \((l - 1)\) are the same as the commodities in the sets \( o'_l, o'_l, o'_l, o'_l \) and that the rank of the paths used by these commodities in \( P'_l \) are larger than or equal to the rank of the paths in suffix \((l - 1)\), we obtain that \( v(o'_l) + v(o'_l) + v(o'_l) + v(o'_l) \leq B_{\text{suffix}}(l - 1) \). Finally we conclude that \( v(P'_l) \leq B_{\text{prefix}}(P_i) + B_{\text{suffix}}(l - 1) = \tilde{B}(P_i) \).

Case 2. \( \tilde{B}(P_i) = B_{\text{prefix}}(P_i) + B_{\text{label}}(P_i) + B_{\text{suffix}}(l) \). Again \( v(P'_l) = v(o'_l) + v(o'_l) + v(o'_l) + v(o'_l) \). As in the first case, we have that \( v(o'_l) \leq B(o'_l) = B_{\text{prefix}}(P_i) \) and \( v(o'_l) + v(o'_l) + v(o'_l) \leq B_{\text{suffix}}(l) \). By the definition of a descendant the rank of the path of commodity \( l \) in \( P_i \) is less than or equal to the rank of the path of commodity \( l \) in \( P'_l \). In other words it means that the cost (associated with zero tariffs) of the path of commodity \( l \) in \( P_i \) is less than or equal to the cost (associated with zero tariffs) of the path of commodity \( l \) in \( P'_l \). Then by the definition of the upper bound on the revenue raised on a ‘label’ commodity, \( B_{\text{label}}(P^l) \) in \( P_i \) is larger than or equal to \( B_{\text{label}}(o'_l) \) in \( P'_l \) and \( B_{\text{label}}(o'_l) \leq v(o'_l) \). In conclusion we have

\[
v(P'_l) \leq B_{\text{prefix}}(P_i) + B_{\text{label}}(P^l) + B_{\text{suffix}}(l).\]

Finally, Proposition 3 assures that the multipaths are explored in decreasing order of their (improved) upper bound \( B(P_i) \) defined recursively as follows.

Proposition 3. Let \( B(P_i) = \tilde{B}(P_i) \) and \( P_i \) be a multipath with label \( l \) and \( P'_l \) a descendant from \( P_i \). Then \( B(P'_l) = \min\{B(P_i), \tilde{B}(P'_l)\} \) is an upper bound on the revenue associated with the multipath \( P'_l \) and \( B(P_i) \geq B(P'_l) \).

6. Stepwise description of the algorithm

The exact multipath algorithm (EMA) is composed of ‘suffix’ and ‘solver’ phases, both of which are based on MRU (iter, k), a function that returns a multipath and a revenue upper bound based on a limited number of commodities (k to \( |K| \)), as described in the two previous sections. In the suffix phase, we have \( k \geq 1 \) and an upper bound on the number of explored multipaths is set to iter. In the solver phase, \( k = 1 \) and iter is set to \( +\infty \).

MRU(iter, k)

Step 0. Initialization

- \( i \leftarrow 1 \).
- \( P^* \leftarrow \text{NULL}, L(P^*) \leftarrow -\infty \).
- \( \text{LIST} \leftarrow P^1 = (p^1_1, \ldots, p^1_{|K|}) \).

Step 1. Multipath selection and evaluation

- Let \( P_i \in \arg \max\{L(P) | P \in \text{LIST}\} \) and \( P^* \leftarrow B(P_i) \).
- Let \( L(P_i) \) be the revenue associated with multipath \( P_i \).
- If \( L(P^*) < L(P_i) \) then \( P^* \leftarrow P_i \).

Step 2. Stopping criteria

- If \( B^* \leq L(P^*) \) then stop. \( (B^*, P^*) \) is optimal for commodities belonging to \([k, \ldots, |K|]\).
- If \( i > \text{iter} \) then stop. \( B^* \) is a valid upper bound associated with multipath \( P^* \) for commodities belonging to \([k, \ldots, |K|]\).

Step 3. LIST update

- Determine the descendants \( P'_l \) of \( P_i \) as in Section 4, and the associated upper bounds using \( B(P'_l) = \min\{B(P_i), \tilde{B}(P'_l)\} \).
- Append the descendants of \( P_i \) with feasible prefix to \( \text{LIST} \).
- \( i \leftarrow i + 1 \).
- Go to Step 1.
Note that at Step 1 of algorithm MRU, there may exist a multipath $P_1$ such that no tariff policy is compatible with it. In such a situation, the corresponding inverse program is infeasible and we set $L(P_1) \leftarrow -\infty$. While it is costly to rule out all such multipaths a priori, one can easily determine whether a single path is compatible with respect to a given commodity considered in isolation. Such quick checks are performed before performing MRU, and all multipaths containing at least one incompatible single path are discarded.

Before defining algorithm EMA, we describe the diversification routine that we implemented in order to improve the lower bound provided by MRU$(\text{Maxiter}, k)$. This routine aims to explore multipaths differing from the current one by the path rank of a commodity. While it is not essential for global convergence of the algorithm, it has a significant impact on the performance of the algorithm at low computational cost. Consequently the diversification phase, called within the Suffix Phase (Step 1) of algorithm EMA, is an integral part of the algorithm.

The diversification routine $\text{DIV}(P, k)$ takes as input a commodity index $k$ and a multipath $P$ and returns a multipath as output. It evaluates via inverse optimization the revenue associated with multipaths belonging to some neighborhood of the current one, and retains the best one. We define the neighborhood associated with multipath $P$, as the set of multipaths obtained by increasing separately by $u$ the rank of each commodity whose index belongs to $[l, |\mathcal{K}|]$, the other ranks being fixed at their value. In other words, the neighborhood $C(u, P_i)$ of integer degree $u$ associated with multipath $P_i$ is defined as:

$$C(u, P_i) = \left\{ (p_1, p_2, \ldots, p_{|\mathcal{K}|}) \in C | \exists k' \in [l, |\mathcal{K}|] \text{ such that } p_k = \begin{cases} p_k^{(k')+u}, & \text{if } k' = k, \\ p_k^{(k)}, & \text{otherwise,} \end{cases} \right\}$$

where $p_k^{(k)}$ is the path with rank $i(k)$ used by commodity $k$. The algorithm halts whenever a prespecified number of the neighboring multipaths have been probed.

In the initialization step, the first multipath $\tilde{P}$ is composed of two parts: the paths in multipath $P$ (defined at the end of MRU$(\text{Maxiter}, k)$) for the commodities belonging to $[k, \ldots, |\mathcal{K}|]$, and the shortest paths corresponding to null tariffs for commodities in $[1, \ldots, k-1]$.

We now proceed with the high level description of the exact algorithm (EMA) followed by the diversification phase $\text{DIV}(P, k)$.

**Exact Multipath Algorithm (EMA)**

**Step 0. Initialization**

- Compute the sets $\mathcal{P}^k$, $k = 1 \ldots |\mathcal{K}|$ using Yen’s algorithm. For each commodity, retain the paths for which there exists a tariff vector compatible with the current path, obtained through inverse optimization.
- Let $P^* \leftarrow \text{NULL}$, $L(P^*) \leftarrow -\infty$
- Set $\text{Maxiter}$ to the maximum number of iterations allowed.

**Step 1. Suffix Phase**

- For $k \leftarrow |\mathcal{K}|$ to 1
  - Let $(B, P) \leftarrow \text{MRU}(\text{Maxiter}, k)$.
  - Let $(P) \leftarrow \text{DIV}(P, k)$.
  - If $L(P^*) < L(P)$ then $P^* \leftarrow P$.
  - $B_{\text{suffix}}(k) \leftarrow B$.

**Step 2. Solver Phase**

- Let $(B, P) \leftarrow \text{MRU}(+\infty, 1)$.
- If $L(P^*) < L(P)$ then $P^* \leftarrow P$.
- $P^*$ is an optimal solution.

**DIV($P, k$)**

**Step 1. Initialization**

- Define $\tilde{u}$, the size of the neighborhood.
- Let $\tilde{P}$ be the extended multipath obtained from $P$ and $L(\tilde{P})$ its value.
- $u \leftarrow 1$.

**Step 2. Neighborhood evaluation**

- While $u \leq \tilde{u}$
  - Let $P^*$ be the best multipath belonging to the neighborhood $C(u, \tilde{P})$.
  - If $L(P^*) > L(\tilde{P})$ then $\tilde{P} \leftarrow P^*$, $u \leftarrow 1$.
  - Else $u \leftarrow u + 1$.
- If $L(P) > L(\tilde{P})$ then $\tilde{P} \leftarrow P$.
Fig. 4. Network structures.

Table 2
Prefix upper bounds.

<table>
<thead>
<tr>
<th>#OD</th>
<th>LMS_LMS</th>
<th></th>
<th></th>
<th></th>
<th>IO_LMS</th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Sol</td>
<td>Gap</td>
<td></td>
<td></td>
<td>Sol</td>
<td>Gap</td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>3183.20</td>
<td>1.85</td>
<td>2568,653.50</td>
<td>3578.97</td>
<td>3183.20</td>
<td>0.00</td>
<td>17,858.20</td>
<td>78.22</td>
</tr>
<tr>
<td>20</td>
<td>4859.00</td>
<td>22.46</td>
<td>2086,527.25</td>
<td>7200.01</td>
<td>4859.00</td>
<td>11.69</td>
<td>899,430.62</td>
<td>7200.01</td>
</tr>
<tr>
<td>30</td>
<td>6096.00</td>
<td>28.12</td>
<td>1336,290.38</td>
<td>7200.01</td>
<td>6096.00</td>
<td>19.07</td>
<td>543,383.19</td>
<td>7200.01</td>
</tr>
<tr>
<td>40</td>
<td>6708.60</td>
<td>36.76</td>
<td>611,609.38</td>
<td>7200.02</td>
<td>6708.60</td>
<td>30.91</td>
<td>376,874.81</td>
<td>7200.01</td>
</tr>
</tbody>
</table>

7. Numerical results

The multipath algorithm has been tested on both randomly generated and real data. The random instances are based on three network topologies illustrated in Fig. 4: Delaunay triangulations, grid networks and Voronoi diagrams. Given a finite set of S points in the plane, Voronoi diagrams partition the plane into polyhedra, each polyhedron containing the set of points that are closest to a given point of S. Note that Delaunay and Voronoi graphs are dual to each other. The grid networks contain 60 nodes (5 × 12) and 208 two-way arcs, while Delaunay and Voronoi graphs were obtained by generating 60 points uniformly on a square (see [12]). The number of origin–destination pairs varies from 10 to 100, and the proportion of tariff arcs from 5% to 20%. Tariff arcs are selected as in [5], with the aim of designing networks with strong interactions across the paths associated with the commodities.

The real life instances [13] correspond, respectively, to a freight transportation network in the Nord-Pas-de-Calais region of Northern France, and to a doctored version of France Telecom’s backbone network.

CPLEX 9.0 has been used in the column generation phase of the algorithm, while shortest path problems were solved using Tarjan’s algorithm [14]. To speed up the column generation procedure, all columns with strictly positive reduced cost were removed whenever that number exceeded 30 × |K|. The method was coded in C++ on a 3 GHz AMD Opteron Processor.

A good value of the parameter $\bar{u}$ for the diversification phase should achieve a trade-off between the size of the neighborhood and the computation time. In the computational experiments it has been set to 5. Note that if $\bar{u} = 1$ the neighborhood is identical with the one described in Section 4.

Before considering the performance of the EMA algorithm itself, we first assess the quality of the improved upper bound with respect to the simple bound considered by Labbé et al. (LMS) [1] and used by Didi Biha et al. [10]. We investigate separately the impact of the prefix upper bound and the recursive suffix upper bound. These experiments were performed on grid networks involving 10–40 commodities, and where the proportion of tariff arcs is fixed at 15%. In Tables 2 and 3, the label ‘#OD’ refers to the number of origin–destination pairs, ‘Gap’ denotes the optimality gap $(Z_{sup} - Z_{inf})/Z_{inf}$, where $Z_{sup}$ is the upper bound and $Z_{inf}$ the best lower bound (or solution) at the end of the algorithm. Column label ‘#it’ denotes the number of iterations of the algorithm. Computation times are reported in seconds in column ‘CPU’ and are limited to 7,200. Each row of the table corresponds to statistics averaged over 5 instances for the corresponding data set.

Computational results reported in Table 2 assess the performance of the prefix upper bounds.

We compare EMA performances under the bounds (LMS) defined by Labbé et al. and the improved bound of Section 5 (IO) for the suffix while keeping LMS for the prefix. The bounds are denoted by: (LMS_LMS) and (IO_LMS). While the objective value of the best solution is identical for both methods, the quality of leader’s revenue upper bounds is better for IO_LMS than for LMS_LMS as illustrated in column Gap.

The results displayed in Table 3 concern the suffix upper bound computation. The algorithm described in Section 6 is denoted IO_SUFF(Maxiter), where Maxiter is the maximal number of iterations of MRU for the upper bound computation. Note that IO_SUFF(0) corresponds to IO_LMS in Table 2. In Table 3 the additional column SUFF represents the computation time required by the suffix upper bound computation.

Considering the instances with 10 origin–destination pairs solved to optimality, the number of iterations of the algorithm #it decreases with Maxiter. Considering the instances unsolved to optimality, the quality of the solutions measured by the
Table 3
Suffix upper bounds.

<table>
<thead>
<tr>
<th>#OD</th>
<th>Gap</th>
<th>#it</th>
<th>CPU</th>
<th>SUFF</th>
<th>Gap</th>
<th>#it</th>
<th>CPU</th>
<th>SUFF</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>IO_SUFF(0)</td>
<td></td>
<td></td>
<td></td>
<td>IO_SUFF(500)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>17,858.20</td>
<td>78.22</td>
<td>–</td>
<td>0.00</td>
<td>3,650.00</td>
<td>25.16</td>
<td>8.18</td>
</tr>
<tr>
<td>20</td>
<td>11.69</td>
<td>899,430.62</td>
<td>7200.01</td>
<td>–</td>
<td>4.25</td>
<td>751,234.38</td>
<td>7200.01</td>
<td>62.68</td>
</tr>
<tr>
<td>30</td>
<td>19.07</td>
<td>543,383.19</td>
<td>7200.01</td>
<td>–</td>
<td>9.72</td>
<td>442,866.00</td>
<td>7200.01</td>
<td>209.45</td>
</tr>
<tr>
<td>40</td>
<td>30.91</td>
<td>376,874.81</td>
<td>7200.01</td>
<td>–</td>
<td>19.44</td>
<td>310,426.59</td>
<td>7200.01</td>
<td>422.75</td>
</tr>
<tr>
<td></td>
<td>IO_SUFF(3000)</td>
<td></td>
<td></td>
<td></td>
<td>IO_SUFF(5000)</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>0.00</td>
<td>3,319.00</td>
<td>27.42</td>
<td>12.04</td>
<td>0.00</td>
<td>3,319.00</td>
<td>27.53</td>
<td>12.12</td>
</tr>
<tr>
<td>20</td>
<td>3.57</td>
<td>737,598.62</td>
<td>7200.01</td>
<td>200.26</td>
<td>3.34</td>
<td>798,337.19</td>
<td>7200.01</td>
<td>274.17</td>
</tr>
<tr>
<td>30</td>
<td>8.66</td>
<td>423,321.00</td>
<td>7200.02</td>
<td>644.87</td>
<td>8.44</td>
<td>399,673.81</td>
<td>7200.01</td>
<td>959.93</td>
</tr>
<tr>
<td>40</td>
<td>17.69</td>
<td>267,324.81</td>
<td>7200.01</td>
<td>1307.54</td>
<td>17.32</td>
<td>234,493.80</td>
<td>7200.01</td>
<td>1979.97</td>
</tr>
</tbody>
</table>

Table 4
Grid networks (1).

<table>
<thead>
<tr>
<th>#OD</th>
<th>%T</th>
<th>MIP+</th>
<th>EMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NOpt</td>
<td>Gap</td>
</tr>
<tr>
<td></td>
<td></td>
<td>NOpt</td>
<td>Gap</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>0.00</td>
<td>131.30</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>4.80</td>
<td>42,538.24</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>9.95</td>
<td>45,620.50</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>9.95</td>
<td>45,564.17</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>2.54</td>
<td>23,959.65</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>30.87</td>
<td>45,378.86</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>31.28</td>
<td>45,231.15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>33.22</td>
<td>45,136.12</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
<td>16.57</td>
<td>45,166.15</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
<td>38.58</td>
<td>44,956.41</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
<td>40.13</td>
<td>44,755.51</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
<td>40.08</td>
<td>44,832.22</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
<td>31.64</td>
<td>44,923.53</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
<td>49.79</td>
<td>44,987.47</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
<td>58.99</td>
<td>44,563.13</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
<td>52.86</td>
<td>44,384.04</td>
</tr>
</tbody>
</table>

ratio Gap tends to improve with Maxiter. CPU times reported in the SUFF column increases linearly with Maxiter. In the next experiments, Maxiter has been set to 3,000. The results in Tables 2 and 3 are self-explanatory. Note that the improvements resulting from the incorporation of the upper bound are crucial in the EMA algorithm.

To assess the quality of the multipath algorithm, its results were contrasted against those obtained by CPLEX 9.0 on the MIP formulation proposed by Labbé et al. [1], where the complementarity constraints are linearized in the standard fashion. Actually, we propose to improve MIP by requiring that the inverse optimization procedure be applied at every node of the B&B tree. This allows us to rapidly isolate a feasible solution and improve the gap when no convergence occurs within the time limit. The resulting improved MIP is labeled MIP+. Computation time was limited to 12 h (43,200 s), although CPLEX may slightly exceed that time limit, due to its CPU time management.

Results are summarized in Tables 4–7. In Table 4 commodities are selected to generate interactions between origin–destination pairs, whereas they are randomly generated in Tables 5–7. In each table and in addition to already defined symbols, column ‘%T’ denotes the percentage of tariff arcs, ‘NOpt’ represents the number of instances solved to optimality and ‘Rinf’ (resp. ‘Rsup’) represents the ratio between lower bound (resp. upper bound) of the EMA algorithm and the lower bound (resp. upper bound) obtained with MIP+ formulation. In each row of the tables, statistics are averaged over 5 instances. As a general rule, the algorithm dominates the MIP+ formulation on all instances for which the percentage of tariff arcs is less than or equal to 15%.

All instances with 10 origin–destination pairs in Tables 5–7 are solved to optimality under MIP+ and EMA. Computation times of EMA are less than or equal to those required by MIP+. Instances involving more than 10 origin–destination pairs remain difficult to solve even for EMA, although the gaps obtained under EMA are lower than those produced by MIP+.

The instances in Table 4 proved much more challenging than those of Table 5 for MIP+. Indeed, while the multipath algorithm succeeds in solving all instances involving 10 commodities, this is not true of MIP+ in Table 4, whose computation times, whenever it could uncover an optimal solution, are significantly larger than those of EMA.

More instances with the Delaunay structure are solved to optimality with EMA than with MIP+. For the remaining one, the gap obtained with EMA is lower than the MIP+ gap.

The Voronoi topology proved more difficult to solve than either grid or Delaunay, with only instances involving 10 commodities or 5% tariff arcs solvable to proven optimality. The smaller node degrees in Voronoi diagrams actually make for stronger interactions between commodities, and hence combinatorially more challenging instances. In networks involving
Table 5
Grid networks (2).

<table>
<thead>
<tr>
<th>#OD</th>
<th>%T</th>
<th>MIP+</th>
<th>EMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NOpt</td>
<td>Gap</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>2</td>
<td>3.90</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>1</td>
<td>10.49</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>0</td>
<td>10.49</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>2</td>
<td>3.26</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0</td>
<td>25.49</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>0</td>
<td>26.12</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>0</td>
<td>21.17</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0</td>
<td>18.72</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0</td>
<td>44.04</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>0</td>
<td>50.81</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>0</td>
<td>53.97</td>
</tr>
</tbody>
</table>

Table 6
Delaunay networks.

<table>
<thead>
<tr>
<th>#OD</th>
<th>%T</th>
<th>MIP+</th>
<th>EMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>NOpt</td>
<td>Gap</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>4</td>
<td>1.38</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>3</td>
<td>1.71</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>4</td>
<td>1.71</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>3</td>
<td>3.06</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0</td>
<td>11.91</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>0</td>
<td>12.34</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>0</td>
<td>15.31</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0</td>
<td>17.87</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0</td>
<td>26.12</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>0</td>
<td>36.88</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>0</td>
<td>37.46</td>
</tr>
</tbody>
</table>

A high density of tariff arcs, the high level of overlap between the paths associated with distinct commodities make for instances where the optimal path structure is harder to identify. On such networks EMA appears to be less efficient.

The real life instances are issued from the freight transportation and telecommunication areas. First we consider an aggregate transportation network of the Nord-Pas-de-Calais region (France) consisting of 78 nodes, 180 links and 40 OD pairs. The transportation network supports a large amount of freight flows mainly due to the locations of three main harbors (Dunkerque, Calais and Boulogne), a large metropolitan area (Lille) and a multi-modal transportation platform in Dourges (road, railway, waterway network). Moreover the region is criss-crossed from north to south and east to west by national or international highways. The transportation data come from the Conseil Régional du Nord-Pas-de-Calais. Three scenarios have been considered, the first (NPDC1) corresponding to the base scenario. The second scenario is characterized by the existence of a direct tariff link between each OD pair (NPDC2) and the third by the increase of freight to and from Dunkerque (NPDC3). The cost of the arcs are related to the distance and the type of the associated road.

Second we consider telecommunication instances. In this realm, the leader consists of a service provider who sets usage prices. The network represents the French telecommunication backbone network (doctored for confidentiality reasons), that consists of respectively 11, 15, 22 nodes, 100, 209, 420 links. Three instances were considered, involving 42, 54 and 80 origin–destination pairs, respectively.

The results, which are reported in Table 8, are self-explanatory, with the complexity of telecommunication instances increasing with the number of OD pairs. Optimal solutions are computed easily by both methods for TELE1 and TELE2. The lower bound obtained by EMA for TELE3 is of higher quality than the lower bound obtained with MIP+. No freight transportation instances could be solved to optimality by MIP+ although NPDC1 and NPDC2 could be solved with EMA within the allocated time limit. The revenue upper bounds obtained with EMA outperform the bounds obtained with MIP+. Instance NPDC2 with a direct tariff link between each OD pair seems easier to solve. From a qualitative point of view, it has been observed that most tariffs are set to values that equalize the cost of the taxable path (whenever it is used) and
Table 7
Voronoi networks.

<table>
<thead>
<tr>
<th>#OD</th>
<th>%T</th>
<th>MIP^+</th>
<th>EMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOpt</td>
<td>Gap</td>
<td>CPU</td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>3</td>
<td>2.13</td>
</tr>
<tr>
<td>25</td>
<td>5</td>
<td>5</td>
<td>0.00</td>
</tr>
<tr>
<td>25</td>
<td>10</td>
<td>0</td>
<td>12.62</td>
</tr>
<tr>
<td>25</td>
<td>15</td>
<td>0</td>
<td>17.45</td>
</tr>
<tr>
<td>25</td>
<td>20</td>
<td>0</td>
<td>9.87</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>0</td>
<td>13.12</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>0</td>
<td>36.40</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>0</td>
<td>25.92</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>0</td>
<td>21.23</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>0</td>
<td>44.23</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>0</td>
<td>66.31</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>0</td>
<td>25.92</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>0</td>
<td>21.23</td>
</tr>
</tbody>
</table>

Table 8
Real life instances.

<table>
<thead>
<tr>
<th>%T</th>
<th>#OD</th>
<th>MIP^+</th>
<th>EMA</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>NOpt</td>
<td>Gap</td>
<td>CPU</td>
</tr>
<tr>
<td>Tele1</td>
<td>35</td>
<td>42</td>
<td>0.00</td>
</tr>
<tr>
<td>Tele2</td>
<td>35</td>
<td>54</td>
<td>0.00</td>
</tr>
<tr>
<td>Tele3</td>
<td>35</td>
<td>80</td>
<td>0.01</td>
</tr>
<tr>
<td>NPDC1</td>
<td>23</td>
<td>40</td>
<td>0.01</td>
</tr>
<tr>
<td>NPDC2</td>
<td>31</td>
<td>40</td>
<td>0.01</td>
</tr>
<tr>
<td>NPDC3</td>
<td>23</td>
<td>40</td>
<td>0.03</td>
</tr>
</tbody>
</table>

the shortest tariff-free path. However it can be observed that, for some OD pairs, tariffs are set to smaller values in order to capture traffic arising from other origin–destination pairs, and such a consolidation is required to maximize the revenue.

8. Conclusion

In this paper, we have introduced and implemented an exact path-based algorithm for a network pricing problem, which proves efficient for tackling instances where the percentage of tariff arcs is small, an assumption that is likely to hold in practical situations. The main feature of the algorithm is the treatment of lower and upper bounds on the value of an optimal solution. Lower bounds are obtained through a decomposition algorithm (Dantzig–Wolfe) while tight upper bounds, which are notoriously difficult to produce for that bilevel program, are derived from a recursive call to the main algorithm. Further work will focus on efficient heuristic procedures able to tackle to near optimality yet larger instances.

References