A Tabu search 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, Canada H2T 2N8
c CIRRELT, Département d’Informatique et de Recherche Opérationnelle, Université de Montréal C.P. 6128, Succursale Centre-Ville, Montréal, QC, Canada H3C 3J7
d GERAD, Département de Mathématiques et de Génie Industriel, Ecole Polytechnique de Montréal C.P. 6079, Succursale Centre-Ville, Montréal, QC, Canada H3C 3A7

A R T I C L E I N F O
Available online 21 January 2012
Keywords:
Tabu Search
Pricing problem on network
Combinatorial optimization

A B S T R A C T
In this paper, we propose an efficient Tabu Search procedure for solving the NP-hard network pricing problem. By exploiting the problem’s features, the algorithm allows the near-optimal solution of problem instances that are out of reach of exact combinatorial methods.
© 2012 Elsevier Ltd. All rights reserved.

1. Introduction

In this paper we address the numerical solution of the network pricing problem (NPP) introduced by Labbé et al. [12] involving two decision makers acting non-cooperatively and in a sequential way. In this context an operator (the leader) sets prices on a subset of arcs of an uncongested transportation network, with the aim of maximizing revenue, explicitly taking into account the reaction of cost-minimizing users (the follower). For fixed prices, user flows are assigned to shortest paths with respect to the generalized cost, i.e., the sum of the original costs and prices. Whenever ties occur among shortest paths, the paths selected are the ones yielding the highest revenue for the leader.

The NPP has been used in the context of airline revenue management [5], and a better understanding of its inner structure is a prerequisite to analyzing extensions of prime interest that would involve features such as congestion, population segmentation, randomness and dynamics.

On the theoretical side, NP-hardness of the NPP has been proved by Labbé et al. [12], a result later refined by Roch et al. [13] and Grigoriev et al. [10]. Special structures have been investigated by Dewez et al. [6], Bouthou et al. [1], Heilporn [11] and, recently, a link between variants of the NPP and more traditional pricing problems in economics. From now on, the focus will be on the basic NPP, without topological restrictions on the network nor constraints on the prices, with the (possible) exception of nonnegativity.

Exact methods for solving the NPP are often based on the characterizations of the lower level shortest paths by their optimality conditions (see [12]). If one linearizes the complementarity slackness conditions, the modified formulation takes the form of a mixed integer program (MIP+) that can then be solved by traditional methods. Alternatively, in the view that revenue maximizing prices compatible with a given lower-level solution can easily be recovered, one can address the NPP as a pure combinatorial problem whose decision variables are origin-destination paths. This approach was first developed by Didi et al. [7] and later extended by Brotcorne et al. [4] who developed an efficient path generation procedure, obtained tight upper bounds for the revenue, and solved the inverse pricing problem by column generation. It has been observed that the path-based approach is more appropriate than the MIP+ one when negative prices are allowed, although none allows the solution of large-scale instances.

On the heuristic front, Brotcorne et al. [2,3] have developed primal-dual heuristics based on a single-level formulation of the NPP. Unfortunately, these do not scale well with the number of commodities (origin-destination pairs), which restricts the applicability of the method.

The current paper is concerned with the development of a Tabu framework and algorithm (TABUPRICE) capable to solve large instances of the NPP, in terms of the number of origin-destination pairs (more than 30 origin-destination pairs). Throughout the algorithm we exploit the tree-like structure of the lower level problem for fixed upper level prices. More precisely, for a given set of user paths, the corresponding leader’s revenue is computed through the inverse optimization procedure introduced in Brotcorne et al. [4].

The rest of the paper is organized as follows. Following the description of the model and algorithmic framework in Section 2, we provide a detailed account of the main components of the algorithm: neighborhood structure and evaluation (Section 3), Tabu phase (Section 4), and diversification phase (Section 5).
This leads to numerical experiments performed on randomly
generated instances corresponding to three topologies (Section 6).

2. Model and algorithmic framework

In this section we first define notations and the bilevel model for
the network pricing problem. Next we describe the algorithmic
framework for the Tabu Search Algorithm (TABUPRICE). 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 prices set by an operator, and \( \mathcal{A}_2 \) its complement \( \mathcal{A} - \mathcal{A}_1 \). Each arc \( (i,j) \in \mathcal{A}_1 \) (respectively \( (i,j) \in \mathcal{A}_2 \)) is endowed with a unit cost \( c_{ij} \) (respectively \( d_{ij} \)). To each origin–
destination pair \( (o(k), d(k)) \) \( (k \in K) \) we associate a commodity \( k \), a
demand \( n_k \) and the nodal demand vector \( b^k \) defined as

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

The network pricing problem (NPP in short) consists of maximizing
the revenue raised from prices \( p_k \) set on arcs \( e \in \mathcal{A}_1 \), 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 prices. It can be mathematically formulated as the bilevel
program

\[
    \text{NPP :} \quad \max \sum_{k \in K} \sum_{(i,j) \in A_1} p_k x^k_{ij} \\
    \text{s.t. for all } k \left( x^k_{ij} + y^k_{ij} \right) - \left( x^k_{ji} + y^k_{ji} \right) = b^k_i, \quad \forall i \in \mathcal{N},
\]

\[
    \left( x^k_{ij} \right) \geq 0, \quad \forall (i,j) \in A_1, \quad \forall i \in \mathcal{N},
\]

\[
    \left( y^k_{ij} \right) \geq 0, \quad \forall (i,j) \in A_2, \quad \forall i \in \mathcal{N}.
\]

In order that the problem be well posed, we make the standard
assumption, as in Labbé et al. [12], that there exists at least one
price-free path for each origin–destination pair and that, for a
given price vector \( p \), the follower will select the path with
minimal cost favoring the leader. A solution of NPP is thus defined
as a price vector and the associated commodities minimum
cost paths.

As in Roch et al. [13] we consider an equivalent combinatorial
version of this problem. To this aim, we replace the lower level
problem of NPP by its primal-dual optimality conditions, to obtain
the bilinear programming formulation

\[
    \text{BILIN :} \quad \max \sum_{k \in K} \sum_{(i,j) \in A_1} p_k x^k_{ij} \\
    \text{s.t. for all } k \left( x^k_{ij} + y^k_{ij} \right) - \left( x^k_{ji} + y^k_{ji} \right) = b^k_i, \quad \forall i \in \mathcal{N},
\]

\[
    x^k_{ij} \geq 0, \quad \forall (i,j) \in A_1, \quad \forall i \in \mathcal{N},
\]

\[
    y^k_{ij} \geq 0, \quad \forall (i,j) \in A_2, \quad \forall i \in \mathcal{N},
\]

\[
    \alpha^k_{ij} - \beta^k_{ij} \leq c_{ij} + p_k, \quad \forall (i,j) \in A_1, \quad \forall i \in \mathcal{N},
\]

\[
    \gamma^k_{ij} - \delta^k_{ij} \leq d_{ij}, \quad \forall (i,j) \in A_2, \quad \forall i \in \mathcal{N}.
\]

This combinatorial structure is amenable to an efficient Tabu
algorithm which proceeds in forest space.

The algorithmic framework follows the traditional approach
outlined, for instance by Gendreau and Potvin [9]. It is composed
of three main phases: local ascent, Tabu search, and diversification
(see Fig. 1). The local ascent phase determines the best
solution within a neighborhood of the current solution. Next, the
algorithm switches to the Tabu phase, where non-improving
moves are allowed, while cycling is prevented through the use
of a Tabu list that records the recent search history. The Tabu
phase is halted whenever an improved solution is found, or a
predefined number of iterations has been reached. In the first

**Fig. 1.** Algorithmic framework.

**Fig. 2.** Spanning tree in the single commodity case.
branch of the alternative, one goes back to local ascent. In the second branch, one enters the diversification phase, which forces the investigation of previously unexplored areas of the search space. The latter is performed for a fixed number of iterations. In case of improvement, it is followed by a local ascent phase. Otherwise, the algorithm reverts to a Tabu phase initiated at the best solution achieved so far.

More specifically, the local ascent and Tabu phases of TABU-PRICE share a common neighborhood structure based on the combinatorial formulation of NPP. Actually, the only difference lies in the update process of the tree-like structures of lower level solutions leading to the next neighborhood evaluation. When local ascent and Tabu phases fail to yield an improved solution, one resorts to diversification, a phase that implements Tabu moves with respect to a perturbed objective. Within diversification, one relies on the long term memory of the process resulting from the insertion of ‘elite’ solutions into a ‘pool’. Such strategy helps to direct the algorithm towards promising areas of the search space. The algorithm stops after a fixed number of iterations, which is highly correlated to the number of origin-destination pairs of the instances.

3. Neighborhood structure and evaluation

The neighborhood structure is the key component of the algorithmic framework. It relies heavily on the tree-like structure of the lower level solution, computed for a given price vector. More precisely, for a fixed price vector, a neighbor of the current solution is obtained by modifying the paths of each commodity while (i) enforcing the tree-like structure, and (ii) ensuring path consistency.1

Let \( \tau_{ok} \) denote a spanning tree rooted at the origin node of commodity \( k \). A neighboring path from \( o(k) \) to \( d(k) \) is derived from \( \tau_{ok} \), through the insertion of a single arc \( e^l = (i,j) \in \tau_{ok} \) into the tree, while ensuring that there exists a subpath \( \tau_{op} \) of \( (o,k) \) in \( \tau_{ok} \) from \( o(k) \) to \( i \), as well as a subpath \( \tau_{p} \) of \( (j,d(k)) \) in \( \tau_{ok} \) from \( j \) to \( d(k) \). A new path \( \tilde{p}_k \) for commodity \( k \) then results from the concatenation of \( \tilde{p}_k = \tau_{op}(o(k),i) \) and \( \tau_{p}(j,d(k)) \).

In the single commodity case, consider the situation illustrated in Fig. 2(a), where commodity \( 1,5 \) is endowed with a unit demand. We set the path used by the follower to \( 1 \rightarrow 3 \rightarrow 2 \rightarrow 5 \) and the current spanning tree to \( \tau_1 \) (see Fig. 2(b)). Since the insertion of arcs \( 1,4 \) or \( 5,1 \) into the spanning tree leaves no feasible path from 1 to 5, whatever the exiting arc, these moves are discarded, which leaves arcs \( 4,5,1,2,3,5 \) as candidate arcs for entering the spanning tree.

In the multi-commodity case, one must take care that, whenever distinct commodities use paths of the spanning tree going through nodes \( u \) and \( v \) (in that order), they actually share the shortest subpath between \( u \) and \( v \). The situation is illustrated in Fig. 3 with commodities \( 1,6 \) and \( 2,6 \). Since both commodities must share identical subpaths between \( u=3 \) and \( v=4 \), this forces the entry of arc \( (4,6) \) into the spanning tree associated with commodity \( 1,6 \), and results in the switch from path \( 1 \rightarrow 3 \rightarrow 5 \rightarrow 6 \) to path \( 1 \rightarrow 3 \rightarrow 4 \rightarrow 6 \) and from path \( 2 \rightarrow 3 \rightarrow 5 \rightarrow 6 \) to path \( 2 \rightarrow 3 \rightarrow 4 \rightarrow 6 \).

In the worst case, the size of the neighborhood is at most equal to the number of commodities times the number of arcs, minus two. To deal with instances involving a large number of commodities, only a subset \( \bar{K} \) of these is considered for neighborhood modification. Initially, the set \( \bar{K} \) contains the \( \bar{K} \) commodities with largest demands. Subsequently, \( \bar{K} \) is composed of the \( K/2 \) commodities yielding the best revenue estimate, while the remaining \( K/2 \) commodities are selected among those that have been least encountered hitherto.

We now consider the problem that consists in evaluating a feasible lower level solution, that is, determining revenue-maximizing prices that are compatible with a given set of commodity paths. This linear program defined by (10) in Section 2 is referred to as an inverse problem. Note that IO may prove infeasible for some moves, and that the corresponding neighbors are discarded. Note also that, using constraints (17) and (18), the objective of IO can be expressed as

\[
\sum_{k \in K} h^k(z^k_{oh} - x^k_{oh}).
\]

The linear programming dual of IO is then

\[
\text{IODUAL : } \min \sum_{k \in K} \sum_{c \in A} c_{c} y_{c}^{k} + \sum_{k \in K} d_{c} z_{c}^{k} \quad \text{s.t. for all } k, \sum_{c \in A} x_{c}^{k} = 0, \forall (i,j) \in A_{1},
\]

\[
\sum_{c \in A} x_{c}^{k} - \sum_{c \in A} x_{c}^{k} = b_{c}^{k}, \forall i \in N',
\]

\[
z_{c}^{k} \geq 0, \forall (i,j) \in A_{1} | x_{c}^{k} = 0,
\]

\[
z_{c}^{k} \geq 0, \forall (i,j) \in A_{2} | y_{c}^{k} = 0,
\]

\[
z_{c}^{k} \text{ free, } \forall (i,j) \in A_{1} | x_{c}^{k} > 0,
\]

\[
z_{c}^{k} \text{ free, } \forall (i,j) \in A_{2} | y_{c}^{k} > 0.
\]

In the single-commodity case, IODUAL reduces to a shortest path problem as shown by Labbé et al. [12]. In the multi-commodity case, IODUAL is a multi-commodity flow problem involving bi-directional arcs corresponding to free variables and the additional constraints (20). This generalized network problem can be solved by an efficient implementation of Dantzig–Wolfe decomposition (see [4]), where the cyclic constraints (21) are viewed as

1 There exist forests that are incompatible with any price vector.
computing shortest path trees associated to these prices. In the Tabu phase however, since the aim is initially to move away from the current solution, a single pivot is performed, yielding a basic solution that might not be compatible with the current price vector.

4. The Tabu phase

As mentioned earlier, the aim of the Tabu phase is to move away from local optima by allowing degradation of the solution with respect to the leader’s objective. This phase is performed whenever no improvement of the current solution is reached through the local ascent phase. The Tabu phase is stopped after a maximum number \( i_{\text{max}} \) of iterations without improvement.

For a given commodity \( k \), let \( e^{\prime} = (i,j) \) denote the arc entering the spanning tree \( \tau_{o(k)} \) and \( e^{\prime} \) the arc leaving \( \tau_{o(k)} \). The move from \( \tau_{o(k)} \) to a neighbor is denoted by \((e^{\prime},e^{\prime},k)\). Cycling among solutions is prevented by considering ‘Tabu’ lists that forbid a number of moves:

- \( L^1 \): list of entering arcs and associated commodity \((e^{\prime},k)\).
- \( L^2 \): list of leaving arcs and associated commodity \((e^{\prime},k)\).
- \( L^3 \): list of pivots (pair of entering and leaving arcs) and associated commodity \((e^{\prime},e^{\prime},k)\).

The first two lists set ‘tags’ that prevent the reversal of past moves, while the third inhibits pivot repetitions. These ‘Tabus’ remain active for a number of iterations \( \theta \) generated randomly and uniformly within the interval \([\underline{\theta},\overline{\theta}]\). A solution is awarded a Tabu status whenever the associated entering arc, leaving arc or pivot is Tabu. If the best solution of the current Tabu phase is revisited in the course of the exploration process, i.e., tags have not been set for a sufficiently long period, then parameters \( \underline{\theta} \) and \( \overline{\theta} \) are incremented by one unit.

As mentioned earlier, modifications to the spanning tree associated with a given commodity may impact other commodities. Whenever this occurs, the Tabu lists are updated and made consistent for all commodities involved at a given move.

Finally, if all neighbors of a current solution are tagged ‘Tabu’, then the simplex pivot \((e^{\prime},e^{\prime})\) that minimizes the penalty function

\[
\pi(e^{\prime},e^{\prime}) = \max\{l^1(e^{\prime}) - 1,0\} + \max\{l^2(e^{\prime}) - 1,0\} + 2 \max\{l^3(e^{\prime},e^{\prime}) - 1,0\}
\]

is performed, where \( l \) is the current iteration index, \( l^1(e^{\prime}) \) denotes the iteration index when arc \( e^{\prime} \) is removed from \( L^1 \), \( l^2(e^{\prime}) \) the iteration index when arc \( e^{\prime} \) is removed from \( L^2 \), and \( l^3(e^{\prime},e^{\prime}) \) the iteration index when pivot \((e^{\prime},e^{\prime})\) is removed from \( L^3 \). The aim of the penalty function is to prevent pivot repetition (hence the factor 2 in the penalty term) while allowing the entry or removal of arc \( e^{\prime} \) or \( e^{\prime} \). At the beginning of the Tabu phase, \( l^1(e^{\prime}), l^2(e^{\prime}) \) and \( l^3(e^{\prime},e^{\prime}) \) are set to zero for all indices and variables.

5. Diversification

The aim of diversification is to explore new regions of the search space, by moving away from current solutions while avoiding strong degradation of the objective. Diversification occurs whenever the Tabu phase fails, i.e., does not improve on the current solution. It is initialized with the best solution \((\hat{x},\hat{y})\) achieved within the Tabu phase.

Diversification is actually nothing else than a Tabu phase involving the modified objective

\[
W((x,y),(\hat{x},\hat{y})) = Z(x,y) + Z(\hat{x},\hat{y}) \times \text{dist}(x,y,(\hat{x},\hat{y})),
\]

Fig. 4. Voronoï network structures. (a) Grid network, (b) Delaunay network and (c) Voronoï network.
commodity paths in good solutions are recorded. If a path is used the pool, statistics concerning the occurrence frequencies of from other members of the pool is sufficiently large. Relying on is encountered, it is included to the pool whenever its distance recorded in an ‘elite’ pool. When a solution that improves revenue of the Ascent or Tabu phases, a fixed number of solutions are term memory is exploited in the following fashion. In the course where $Z(x,y)$ denotes the optimal revenue associated with the solution $(x,y)$, and $\text{diss}(x,y), (\tilde{x}, \tilde{y}))$ is a dissimilarity measure defined as the ratio of the number of different paths among the two solutions over the total number of commodities. Maximizing $W((x,y), (\tilde{x}, \tilde{y}))$ achieves a trade-off between the quality of a solution and its distance from the current point $(\tilde{x}, \tilde{y})$.

Next, in order to improve the diversification process, long-term memory is exploited in the following fashion. In the course of the Ascent or Tabu phases, a fixed number of solutions are recorded in an ‘elite’ pool. When a solution that improves revenue is encountered, it is included to the pool whenever its distance from other members of the pool is sufficiently large. Relying on the pool, statistics concerning the occurrence frequencies of commodity paths in good solutions are recorded. If a path is used by over 90% of the elite solutions, then such path is deemed ‘good’ and stays fixed throughout the remainder of the diversification phase.

The diversification phase is stopped after a maximum number of iterations without improvement of the modified objective.

### 6. Numerical results

TABUPRICE has been tested on randomly generated instances based on three network topologies illustrated in Fig. 4: Delaunay triangulations, grids, and Voronoi networks. Given a finite set of $S$ points in the plane, a Voronoi graph partitions the plane into polyhedra, each polyhedron containing the set of points that are

![Fig. 5](image1.png) Evolution of the TP and exact solution methods on an instance of Brotcorne et al. with 30 commodities, 15% of arcs subject to prices, $p \geq 0$.

![Fig. 6](image2.png) Evolution of the TP and exact solution methods on an instance of Brotcorne et al. with 30 commodities, 15% of arcs subject to prices, $p$ free.
closest to a given point of $S$; Delaunay and Voronoi graphs are dual to each other. Delaunay graphs are well suited for representing telecommunication networks, while grids and Voronoi networks are better suited to the representation of road networks.

The grid networks contain 60 nodes $(5 \times 12)$ and 208 two-way arcs, while Delaunay and Voronoi graphs were obtained by generating 60 points uniformly on a square (see Fortune [15]). The number of origin–destination pairs varies from 10 to 100, and the proportion of priced arcs from 5% to 20%. Priced arcs are selected as in Brotcorne et al. [2], with the aim of designing networks with strong interactions across the paths associated with the commodities. CPLEX 9.0 has been used in the column generation phase of the algorithm, while shortest path problems were solved using Tarjan’s algorithm [16]. TABUPRICE was implemented in C++ on a 3 GHz AMD Opteron Processor.

Prior to conducting the numerical experiments reported in this paper, a number of tests were performed to determine good values of the parameters of the algorithm, through a training set. We only report here the calibration of two important parameters, namely the maximal number of unsuccessful iterations $l_{\text{max}}$ in the Tabu Phase (before resorting to diversification), and the number of commodities $K$ considered for each neighborhood evaluation. These tests involved grid networks where the number of commodities was in the range $(10, 40)$ and the percentage of priced arcs in the range $(5\%, 20\%)$. Parameters $[0, 7]$ were fixed to $[3, 8]$ and $l_{\text{max}}$ to 500.

In Tables 1 and 2, labels #OD and %p refer to the number of origin–destination pairs and percentage of priced arcs, respectively, ‘% denotes the ratio $Z/Z_{\text{inf}}$, where $Z$ is the value of the best solution found in the Tabu phase, and $Z_{\text{inf}}$ is the best lower bound (or solution) achieved by the exact method proposed by Labbé et al. [12]. In the latter, computing time was limited to 2 h, and an asterisk marks those instances for which the algorithm hit this limit before finding an optimal solution, or proving the optimality thereof. Computing times are reported in seconds under the heading CPU. Each row of the table corresponds to statistics averaged over five instances of the corresponding data set.

To assess parameter $l_{\text{max}}$, we compared the results obtained by TABUPRICE with the value of $l_{\text{max}}$ set to $|K|/2$, $|K|$, $1.5|K|$, and $|K|$, and $+\infty$ (see Table 1), the latter value corresponding to the case where diversification never occurs. Throughout, the cardinality of the elite pool was set to 7, while the number of distinct paths required to be inserted into the elite solution pool was set to 2. We observed that, whenever $l_{\text{max}}$ became large, the search process tended to be restricted to a limited number of areas of the search domain (see examples involving 10 or 20 commodities) and $l_{\text{max}} = +\infty$. At the other end of the spectrum, small $l_{\text{max}}$ values prevent a thorough exploration of the current neighborhood (see examples involving 10 or 20 commodities and $l_{\text{max}}$ set to $|K|/2$).

As a general rule, the quality of the solution is only mildly influenced by the value of the parameter $K$, the number of

### Table 3

<table>
<thead>
<tr>
<th>#OD</th>
<th>%p</th>
<th>p</th>
<th>NOpt</th>
<th>%</th>
<th>CPU</th>
<th>Min%</th>
<th>%</th>
<th>CPU</th>
<th>*it</th>
<th>*CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>≥0</td>
<td>5</td>
<td>0.00</td>
<td>0.75</td>
<td>97.91</td>
<td>99.34</td>
<td>2.17</td>
<td>144.98</td>
<td>0.65</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>≥0</td>
<td>5</td>
<td>0.00</td>
<td>3.70</td>
<td>95.92</td>
<td>99.76</td>
<td>3.03</td>
<td>82.21</td>
<td>0.56</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>≥0</td>
<td>5</td>
<td>0.00</td>
<td>30.98</td>
<td>93.24</td>
<td>98.13</td>
<td>4.29</td>
<td>83.22</td>
<td>0.74</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>≥0</td>
<td>5</td>
<td>0.00</td>
<td>170.67</td>
<td>97.72</td>
<td>99.53</td>
<td>6.13</td>
<td>113.45</td>
<td>1.40</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
<td>≥0</td>
<td>5</td>
<td>0.00</td>
<td>4.66</td>
<td>99.09</td>
<td>99.86</td>
<td>9.81</td>
<td>158.47</td>
<td>1.54</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
<td>≥0</td>
<td>5</td>
<td>0.00</td>
<td>311.92</td>
<td>100.00</td>
<td>100.00</td>
<td>15.75</td>
<td>274.72</td>
<td>4.50</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
<td>≥0</td>
<td>4</td>
<td>0.91</td>
<td>16.689.36</td>
<td>99.98</td>
<td>100.00</td>
<td>23.62</td>
<td>447.27</td>
<td>10.44</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
<td>≥0</td>
<td>1</td>
<td>5.79</td>
<td>38.975.47</td>
<td>100.00</td>
<td>100.53</td>
<td>33.65</td>
<td>220.79</td>
<td>7.44</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>EMA</th>
<th>TS</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
</tr>
<tr>
<td>20</td>
<td>5</td>
</tr>
<tr>
<td>20</td>
<td>10</td>
</tr>
<tr>
<td>20</td>
<td>15</td>
</tr>
<tr>
<td>20</td>
<td>20</td>
</tr>
<tr>
<td>30</td>
<td>5</td>
</tr>
<tr>
<td>30</td>
<td>10</td>
</tr>
<tr>
<td>30</td>
<td>15</td>
</tr>
<tr>
<td>30</td>
<td>20</td>
</tr>
<tr>
<td>40</td>
<td>5</td>
</tr>
<tr>
<td>40</td>
<td>10</td>
</tr>
<tr>
<td>40</td>
<td>15</td>
</tr>
<tr>
<td>40</td>
<td>20</td>
</tr>
</tbody>
</table>
commodities in the neighborhood evaluation, although computing times increase linearly with this parameter. In the next experiments, $K$ are set to $0.25 \times |X|$. To assess the efficiency of TABUPRICE, its performance was compared with that of the best known exact methods. In the case where prices are restricted to nonnegative values, CPLEX 9.0 was

<table>
<thead>
<tr>
<th>#OD</th>
<th>$%p$</th>
<th>$p$</th>
<th>NOpt</th>
<th>$%$</th>
<th>CPU</th>
<th>Min%</th>
<th>$%$</th>
<th>CPU</th>
<th>*it</th>
<th>*CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>MIP *</td>
<td>TS</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>5</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>0.21</td>
<td>96.25</td>
<td>99.72</td>
<td>1.59</td>
<td>53.41</td>
<td>0.20</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>0.49</td>
<td>100.00</td>
<td>100.00</td>
<td>2.06</td>
<td>50.17</td>
<td>0.22</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>0.98</td>
<td>100.00</td>
<td>100.00</td>
<td>2.68</td>
<td>200.30</td>
<td>1.33</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>1.83</td>
<td>97.69</td>
<td>99.30</td>
<td>3.34</td>
<td>123.12</td>
<td>0.82</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>44.50</td>
<td>100.00</td>
<td>100.00</td>
<td>63.83</td>
<td>506.10</td>
<td>12.44</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>5918.32</td>
<td>100.00</td>
<td>100.00</td>
<td>145.76</td>
<td>566.50</td>
<td>33.59</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>≥ 0</td>
<td>0</td>
<td>7.65</td>
<td>43.66327</td>
<td>100.17</td>
<td>100.74</td>
<td>183.67</td>
<td>669.16</td>
<td>51.94</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>≥ 0</td>
<td>5</td>
<td>0.00</td>
<td>208.72</td>
<td>96.79</td>
<td>99.34</td>
<td>308.99</td>
<td>885.09</td>
<td>50.94</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>≥ 0</td>
<td>1</td>
<td>11.68</td>
<td>41.12873</td>
<td>99.94</td>
<td>101.03</td>
<td>587.97</td>
<td>1692.94</td>
<td>205.75</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>≥ 0</td>
<td>0</td>
<td>18.46</td>
<td>43.47172</td>
<td>99.75</td>
<td>102.32</td>
<td>858.42</td>
<td>1802.75</td>
<td>283.33</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>≥ 0</td>
<td>0</td>
<td>23.90</td>
<td>43.41458</td>
<td>101.77</td>
<td>105.19</td>
<td>1190.63</td>
<td>1785.24</td>
<td>375.75</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>EMA</th>
<th>TS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>1.71</td>
<td>100.00</td>
<td>100.00</td>
<td>1.80</td>
<td>119.54</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>2.70</td>
<td>99.77</td>
<td>99.95</td>
<td>2.65</td>
<td>183.89</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>7.47</td>
<td>99.58</td>
<td>99.88</td>
<td>3.68</td>
<td>71.97</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>16.47</td>
<td>99.80</td>
<td>99.80</td>
<td>5.23</td>
<td>181.71</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>Free</td>
<td>4</td>
<td>2.58</td>
<td>16.07086</td>
<td>99.38</td>
<td>99.91</td>
<td>85.88</td>
<td>426.09</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>Free</td>
<td>0</td>
<td>16.47</td>
<td>43.20000</td>
<td>101.33</td>
<td>106.59</td>
<td>151.19</td>
<td>674.92</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>Free</td>
<td>0</td>
<td>19.67</td>
<td>43.21159</td>
<td>100.92</td>
<td>105.64</td>
<td>234.43</td>
<td>770.26</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>Free</td>
<td>0</td>
<td>19.40</td>
<td>43.20589</td>
<td>100.71</td>
<td>107.02</td>
<td>359.81</td>
<td>1190.63</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>Free</td>
<td>2</td>
<td>7.67</td>
<td>32.29267</td>
<td>92.28</td>
<td>102.38</td>
<td>412.90</td>
<td>1092.28</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>Free</td>
<td>0</td>
<td>29.00</td>
<td>43.20000</td>
<td>101.52</td>
<td>106.38</td>
<td>850.57</td>
<td>807.23</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>Free</td>
<td>0</td>
<td>32.40</td>
<td>43.20002</td>
<td>101.10</td>
<td>109.66</td>
<td>1424.08</td>
<td>2282.19</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>Free</td>
<td>0</td>
<td>26.54</td>
<td>43.20026</td>
<td>101.04</td>
<td>105.25</td>
<td>2234.62</td>
<td>2502.20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th>EMA</th>
<th>TS</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>1.71</td>
<td>100.00</td>
<td>100.00</td>
<td>1.80</td>
<td>119.54</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>2.70</td>
<td>99.77</td>
<td>99.95</td>
<td>2.65</td>
<td>183.89</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>7.47</td>
<td>99.58</td>
<td>99.88</td>
<td>3.68</td>
<td>71.97</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>16.47</td>
<td>99.80</td>
<td>99.80</td>
<td>5.23</td>
<td>181.71</td>
</tr>
<tr>
<td>50</td>
<td>5</td>
<td>Free</td>
<td>4</td>
<td>2.58</td>
<td>16.07086</td>
<td>99.38</td>
<td>99.91</td>
<td>85.88</td>
<td>426.09</td>
</tr>
<tr>
<td>50</td>
<td>10</td>
<td>Free</td>
<td>0</td>
<td>16.47</td>
<td>43.20000</td>
<td>101.33</td>
<td>106.59</td>
<td>151.19</td>
<td>674.92</td>
</tr>
<tr>
<td>50</td>
<td>15</td>
<td>Free</td>
<td>0</td>
<td>19.67</td>
<td>43.21159</td>
<td>100.92</td>
<td>105.64</td>
<td>234.43</td>
<td>770.26</td>
</tr>
<tr>
<td>50</td>
<td>20</td>
<td>Free</td>
<td>0</td>
<td>19.40</td>
<td>43.20589</td>
<td>100.71</td>
<td>107.02</td>
<td>359.81</td>
<td>1190.63</td>
</tr>
<tr>
<td>100</td>
<td>5</td>
<td>Free</td>
<td>2</td>
<td>7.67</td>
<td>32.29267</td>
<td>92.28</td>
<td>102.38</td>
<td>412.90</td>
<td>1092.28</td>
</tr>
<tr>
<td>100</td>
<td>10</td>
<td>Free</td>
<td>0</td>
<td>29.00</td>
<td>43.20000</td>
<td>101.52</td>
<td>106.38</td>
<td>850.57</td>
<td>807.23</td>
</tr>
<tr>
<td>100</td>
<td>15</td>
<td>Free</td>
<td>0</td>
<td>32.40</td>
<td>43.20002</td>
<td>101.10</td>
<td>109.66</td>
<td>1424.08</td>
<td>2282.19</td>
</tr>
<tr>
<td>100</td>
<td>20</td>
<td>Free</td>
<td>0</td>
<td>26.54</td>
<td>43.20026</td>
<td>101.04</td>
<td>105.25</td>
<td>2234.62</td>
<td>2502.20</td>
</tr>
</tbody>
</table>

Table 4
TS numerical results on grid networks.

Table 5
TS numerical results on Delaunay instances.
applied to the improved \( MIP^+ \) formulation \( MIP^+ \) proposed by Labbé et al. [12], where the complementarity constraints are linearized in the standard fashion, and where inverse optimization is applied at every node of the Branch-and-Bound tree. In the absence of sign restrictions on prices, the exact multipath algorithm (EMA) proposed by Brotcorne et al. [4] was used for comparison. In all cases, computing time was limited to 12 h. Any larger number reported in the tables is due to management of the computer clock by the operating system.

In Figs. 5 and 6, one can observe the progression of the algorithms for instances with and without nonnegativity constraints, respectively, plotted against a logarithmic scale. On a linear scale, the 'variance' increases with the number of commodities in the case of Voronoi topologies. Quite surprising, the behavior of TABUPRICE was not influenced significantly by either sign restrictions on the prices or the network's topology. Although TABUPRICE's CPU time is sensitive to the number of commodities, it is less so than the exact method. Indeed, for instances involving 100 commodities, TABUPRICE's revenues exceeded by 1–10% those of the exact method, while requiring much lower computing times (2300 vs. 43 200).

With respect to another criterion, it is reassuring to observe that, for a given group of problems with similar parameters, the difference between the best and worst performance of TABUPRICE is small, less than 4% for instances involving less than 50 commodities, and less than 7% for instances involving 100 commodities (Delaunay networks and grids). We also observed that this 'variance' increases with the number of commodities in the case of Voronoi topologies. Quite surprising, the behavior of TABUPRICE was not influenced significantly by either sign restrictions on the prices or the network's topology.

### Table 6

<table>
<thead>
<tr>
<th>#OD</th>
<th>( %p )</th>
<th>( p )</th>
<th>NOpt</th>
<th>%</th>
<th>CPU</th>
<th>Min%</th>
<th>%</th>
<th>CPU</th>
<th>*it</th>
<th>*CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>(&lt; 0)</td>
<td>5</td>
<td>0.00</td>
<td>0.67</td>
<td>100.00</td>
<td>1.93</td>
<td>31.59</td>
<td>0.14</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>(&lt; 0)</td>
<td>5</td>
<td>0.00</td>
<td>2.11</td>
<td>99.54</td>
<td>99.91</td>
<td>2.42</td>
<td>29.24</td>
<td>0.19</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>(&lt; 0)</td>
<td>5</td>
<td>0.00</td>
<td>11.10</td>
<td>99.79</td>
<td>99.96</td>
<td>2.90</td>
<td>128.21</td>
<td>0.87</td>
</tr>
</tbody>
</table>

### Table 7

<table>
<thead>
<tr>
<th>#OD</th>
<th>( %p )</th>
<th>( p )</th>
<th>NOpt</th>
<th>%</th>
<th>CPU</th>
<th>Min%</th>
<th>%</th>
<th>CPU</th>
<th>*it</th>
<th>*CPU</th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>5</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>2.77</td>
<td>97.80</td>
<td>99.62</td>
<td>2.78</td>
<td>155.47</td>
<td>0.91</td>
</tr>
<tr>
<td>10</td>
<td>10</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>20.56</td>
<td>99.68</td>
<td>99.92</td>
<td>3.68</td>
<td>236.52</td>
<td>1.77</td>
</tr>
<tr>
<td>10</td>
<td>15</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>12.98</td>
<td>99.68</td>
<td>99.92</td>
<td>3.68</td>
<td>236.52</td>
<td>1.77</td>
</tr>
<tr>
<td>10</td>
<td>20</td>
<td>Free</td>
<td>5</td>
<td>0.00</td>
<td>33.00</td>
<td>98.45</td>
<td>99.33</td>
<td>5.05</td>
<td>193.65</td>
<td>2.18</td>
</tr>
</tbody>
</table>

### Conclusion

By exploiting the structure of the network pricing problem, we developed a Tabu-based method that yielded the best results...
obtained so far on this difficult combinatorial problem. Encouraged by these results, we envision to extend this methodology to variants of the original problem involving behaviorly distinct user classes, as well as randomness of travel delays.

References