# A path-based double projection method for solving the asymmetric traffic network equilibrium problem*

Barbara Panicucci† Massimo Pappalardo† Mauro Passacantando†

**Abstract.** In this paper we propose a new iterative method for solving the asymmetric traffic equilibrium problem when formulated as a variational inequality whose variables are the path flows. The path formulation leads to a decomposable structure of the constraints set and allows us to obtain highly accurate solutions. The proposed method is a column generation scheme based on a variant of the Khobotov's extragradient method for solving variational inequalities. Computational experiments have been carried out on several networks of a medium-large scale. The results obtained are promising and show the applicability of the method for solving large-scale equilibrium problems.

**Keywords:** asymmetric traffic network, equilibrium flow, extragradient method, column generation.

**2000 Mathematics Subject Classifications:** 49J40, 65K10, 90B20.

## 1 Introduction

The purpose of this paper is to provide a method for solving the well known asymmetric traffic network equilibrium problem. In the literature, this problem is modelled as a variational inequality [6, 31]. In particular, there are two main formulations. In one, the variational inequality is expressed in terms of flows on paths of the network; in the other, the variational inequality is expressed in terms of flows on the arcs of the network. The solution methods can thus be broadly divided into two categories according to the solution space the algorithm operates in. The traditional approaches to solving the problem have been arc-based algorithms. Path-formulation has also been considered, for example in [3, 4, 5, 7].

We believe the formulation with path-flow variables to be preferable for several reasons. A major one is that solving the traffic assignment problem in the path flow space automatically provides not only the equilibrium flow, but also all the routed paths. Arc flow based algorithms need, instead, to be supplied with a procedure to yield an equilibrium path flow. On the other hand, there are many applications were path flow solutions are needed as input, such as origin/destination (for short, O/D) matrix estimation and exhaust fume emission analysis [17].

It is also necessary to formulate and solve the traffic assignment problem in the space of path flows when the path costs are non-additive. There are many cases in which it is important to relax the assumption of additive path costs; one such case occurs when users estimate that the time required to travel a given

*This work has been supported by the National Research Program FIRB/RBNE01WBBBB on Large Scale Nonlinear Optimization. This paper has been published in Optimization Letters, vol. 1 (2007), pp. 171–185.

†Department of Applied Mathematics - University of Pisa, via Bonanno 25/b, 56126 Pisa - Italy, e-mail: panicucc@mail.dm.unipi.it.

path increases nonlinearly (e.g. logarithmically) in proportion to the amount of additional traffic [12]. For many types of path cost functions, it is simply impossible to use arc flow variables to formulate and solve the corresponding traffic assignment problem [12].

Another important reason for using path flow variables is that it allows us to determine the equilibrium solution to any given accuracy. This can be done by using the Wardrop equilibrium conditions [2, 35]. In fact, to show that the assignment results are correct, we provide the equilibrium path flows and their relative costs, and observe that the costs on all used paths are equal, and less than the costs on any unused paths. Of course this requires the knowledge of which paths are being used in the network, and the arc-based algorithms do not provide this information. In earlier applications of the traffic assignment problem, an accurate estimation of the arc flow solution was not a main objective. Recently, attention has been directed to Intelligent Transportation Systems (ITS) [14] as part of the strategy for improving the operational safety, efficiency, and security of highway networks. The advent of ITS has made it necessary to find accurate solutions.

From a computational point of view, the simple structure of path representation allows us to decompose the constraints set, and this turns out to be useful for solving large-scale problems.

Furthermore, when path variables are used, it is not necessary to completely enumerate, a priori, all the paths since that would be too expensive computationally even for moderately-sized networks. In fact, there is a standard technique (column generation) which allows us to explicitly identify only those paths that would likely be used. In [18] the authors recognized the utility of embedding a column generation procedure in the general equilibration algorithm proposed in [7], in order to generate paths as needed.

There has been extensive work done on arc-based models for asymmetric equilibrium problems. Several algorithms have been developed for their solution and the relative computational results have been presented (see e.g., [19, 23, 24, 27]). Some numerical results for solving asymmetric equilibrium problems based on path-formulation have been presented in [3, 24, 25, 26, 30]). In the following, we propose an iterative method which allows us to solve, to any given accuracy, large-scale asymmetric traffic equilibrium problems.

The paper is organized as follows: in Section 2 we recall the path-formulation of the traffic network equilibrium problem, in Section 3 we describe the algorithm, then in Section 4 we give a detailed discussion of its implementation, and finally in Section 5 we report the computational results.

## 2 The path-formulated traffic network equilibrium problem

The usual model of a traffic network is given by a direct graph, consisting of a set of nodes $\mathcal{N}$, a set of arcs $\mathcal{A}$, and a set $\mathcal{W}$ of O/D pairs. For each $w \in \mathcal{W}$ there is a known demand $d_w > 0$ representing the rate of traffic entering and exiting the network at the origin and the destination of $w$ respectively. The demand $d_w$ is to be distributed among the paths connecting the O/D pair $w$, the set of which is denoted by $\mathcal{P}_w$. Let $F_p$ denote the portion of $d_w$ routed on path $p$ and let $F$ be the vector of path flows $F_p$, with $p \in \mathcal{P}_w$ and $w \in \mathcal{W}$. We denote the set of feasible path flow vectors by

$$X = \left\{ F \geq 0 : \quad \sum_{p \in P_w} F_p = d_w, \qquad \forall\ w \in \mathcal{W} \right\}. \tag{1}$$

The flow $f_a$ on each arc $a$ is the sum of all flows on paths to which the arc belongs. The arc flow vector, $f = (f_a)_{a \in \mathcal{A}}$, is then given by $f = \Delta\, F$, where $\Delta$ is the arc-path incidence matrix:

$$\Delta_{a,p} = \begin{cases} 1 & \text{if arc } a \in p, \\ 0 & \text{otherwise.} \end{cases}$$

For each arc $a$, there is a non-negative arc cost function $c_a(f)$, which represents the delay in traversing arc $a$ and depends upon the arc flow vector $f$. The corresponding path cost function is assumed to be additive; that is, the travel time $C_p(F)$ on path $p$ is the sum of the travel times on the arcs belonging to $p$:

$$C_p(F) = \sum_{\text{arcs } a \in p} c_a(\Delta F),$$

and thus the path cost vector function is $C(F) = \Delta^T c(\Delta F)$. According to the Wardrop equilibrium principle, the traffic network equilibrium problem is to find a path flow vector $F^* \in X$ which consists of path flows that are positive only on paths with minimum cost. It is well known [6, 31] that this problem is equivalent to the following variational inequality problem (in short VI), which consist of finding a vector $F^* \in X$ such that

$$\langle C(F^*), F - F^* \rangle \geq 0, \qquad \forall\ F \in X, \tag{2}$$

where $\langle \cdot, \cdot \rangle$ denotes the inner product.

# 3 A double projection algorithm for solving the asymmetric traffic equilibrium problem

Problem (2) can be solved by any general computational technique developed for VI. However, we can exploit the structure of the constraints set to get more efficient procedures for solving the equilibrium problem. Since we operate in the space of path flows, the feasible set $X$ is the Cartesian product of as many simplices as the O/D pairs, i.e.

$$X = \prod_{w \in \mathcal{W}} X_w, \qquad \text{where} \qquad X_w = \left\{ F \in \mathbb{R}_+^{|\mathcal{P}_w|} : \sum_{p \in \mathcal{P}_w} F_p = d_w \right\}.$$

Due to the decomposable nature of the constraints set $X$, projection-type algorithms turn out to be easy to apply. Indeed the projection on $X$ of a vector $v$ can be decomposed into a collection of small-sized projections of $v$ on the simplices $X_w$, that is, one projection for each O/D pair. Then, since projections onto simplices $X_w$ can be efficiently computed [22], projection-type methods are a natural solution strategy for solving large-scale VI and have been used, for example, in [3, 5, 25, 26]. Among the projection-type schemes appearing in the literature, the extragradient method [15] is very popular. There are two main reasons for using the extragradient method as a solution algorithm for VI. The first reason is that this method is convergent under mild assumptions on the cost operator. The second is that we do not need, a priori, to know the Lipschitz constant of the cost operator (see Definition 3.1). It should be remarked that, among the projection-type schemes, the hyperplane projection method [32] performs better than the extragradient method for general VI with no special structure. However, since the hyperplane projection method requires the projection onto the intersection between $X$ and a suitable hyperplane, we could not exploit the inherent Cartesian product structure of the feasible set $X$. The specific features of the asymmetric traffic equilibrium problem justify the choice of the extragradient method instead of the hyperplane projection. Moreover, in order to improve the convergence of the extragradient method, we propose a refinement by introducing a new step. The details of the modified extragradient method for solving VI can be described as follows:

**Basic Algorithm for VI**

Step 0. (Initialization) Select parameters $\beta, \xi \in (0,1)$ and $\overline{\alpha} > 0$. Let $F^0 \in X$, $\alpha_0 = \overline{\alpha}$ and set $k = 0$.

Step 1. (Stopping criterion) If a stopping criterion is satisfied then STOP.

Step 2. (Flow update)

**a.** ($\overline{F}^k$ computation) Compute

$$\overline{F}^k = \mathrm{Proj}_X(F^k - \alpha_k\, C(F^k)).$$

**b.** ($\alpha_k$ check)

**while** $\alpha_k > \beta\, \dfrac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|}$ **do**

reduce $\alpha_k$:

$$\alpha_k = \min\left\{\xi\,\alpha_k,\ \beta\, \frac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|}\right\}. \tag{3}$$

**end**

**c.** (Compute $F^{k+1}$ and re-initialize $\alpha_k$) Compute

$$F^{k+1} = \mathrm{Proj}_X(F^k - \alpha_k\, C(\overline{F}^k)),$$

set

$$\alpha_{k+1} = \min\left\{\overline{\alpha},\ \beta \frac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|}\right\}, \tag{4}$$

set $k = k + 1$ and go to step 1.

Here, $\mathrm{Proj}_X(\cdot)$ denotes the Euclidean projection map onto $X$, $F^k$ is the path flow vector at step $k$ and $\alpha_k$ is a positive stepsize. It is known that the convergence rate of Khobotov's algorithm can be very slow. In fact, if at some iteration $\alpha_k$ becomes small, it remains small at all successive iterations. We emphasize that the introduction of a re-initialization of the value of $\alpha_k$ in (4) enables us to avoid this drawback.

This algorithm is proved to converge to an equilibrium flow under some mild assumptions on the cost operator. To make this precise, we introduce the following definitions:

**Definition 3.1.** $C$ is Lipschitz continuous with constant $L$ on $X$ if there exists a positive constant $L$ such that

$$\|C(F_1) - C(F_2)\| \le L\,\|F_1 - F_2\|, \qquad \forall\ F_1, F_2 \in X.$$

**Definition 3.2.** $C$ is pseudomonotone on $X$ if for all $F_1, F_2 \in X$ :

$$\langle C(F_2), F_1 - F_2\rangle \ge 0 \qquad \Longrightarrow \qquad \langle C(F_1), F_1 - F_2\rangle \ge 0.$$

Next we establish the convergence result.

**Theorem 3.1.** If $C$ is pseudomonotone and Lipschitz continuous on $X$, then any cluster point of the sequence $\{F^k\}_{k\in\mathbb{N}}$ is an equilibrium flow.

**Proof.** With our choice for the stepsize, we have:

$$0 < \alpha_k \leq \min \left\{ \overline{\alpha},\, \beta \frac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|} \right\} \qquad \forall\ k \in \mathbb{N}.$$

The proof of convergence follows from [15]. □

As said above, when solving the network equilibrium problem, the first difficulty we meet is the large number of path flow variables, a number which generally grows exponentially in the size of the network. However, computational results (see Table 2, Section 5) show that the number of path variables that are positive in the solution is often very small. Then, a solution strategy is to generate algorithmically, for each O/D pair, only those paths which potentially carry a positive flow in an equilibrium solution (column generation approach). We achieve this goal as follows: we choose a feasible flow vector and we consider the corresponding set of used paths $\mathcal{P}_w^0$, for each O/D pair $w$. Next, at the $k$-th iteration, we compute link costs with respect to the current flow vector, by applying the cost formulas. Using these arc costs, we calculate a shortest path for each O/D pair $w$, which will be added to the set of paths previously obtained, $\mathcal{P}_w^k \subseteq \mathcal{P}_w$, if it has not already been included. Therefore, at the $k$-th iteration, the feasible set $X$ in (1) can be replaced with the following:

$$X^k = \prod_{w \in \mathcal{W}} X_w^k,$$

where

$$X_w^k = \left\{ F \in \mathbb{R}_+^{|\mathcal{P}_w^k|} : \sum_{p \in \mathcal{P}_w^k} F_p = d_w \right\}.$$

The flow is then updated by an iteration of the algorithm for VI previously described. An important feature of this procedure is that, when updating the flow, each projection on the set of feasible path flows $X^k$ can be decomposed into a collection of smaller projections on $X_w^k$, for each pair $w \in \mathcal{W}$.

To accurately describe the specific algorithm developed for solving the asymmetric traffic network equilibrium problem, we need to introduce some notation. At the $k$-th iteration we consider the path flow $F_w^k = (F_p^k)_{p \in \mathcal{P}_w^k}$ and the path cost $C_w(F^k) = (C_p(F^k))_{p \in \mathcal{P}_w^k}$, with $\mathcal{P}_w^k \subseteq \mathcal{P}_w$, for each O/D pair $w$. We denote the flow vector by $F^k = (F_w^k)_{w \in \mathcal{W}}$ and the cost vector by

$$C(F^k) \ = (C_w(F^k))_{w \in \mathcal{W}}.$$

Moreover, we denote by $\mathcal{O} = \{o_1, \ldots, o_r\}$ the set of origin nodes of cardinality $r$. Based on the above discussion, the details of the algorithm can be formalized as follows:

**Algorithm for Traffic Assignment**

Step 0. (Initialization) Select parameters $\beta$, $\xi \in (0,1)$, $\overline{\alpha} > 0$ and a tolerance $\varepsilon > 0$. Let $F^0$ be any feasible path flow. For each pair $w \in \mathcal{W}$, let $\mathcal{P}_w^0$ be the set of used paths. Set $\alpha_0 = \overline{\alpha}$ and $k = 0$.

Step 1. (Column generation) Compute arcs cost with the current path flow $F^k$.

For each pair $w \in \mathcal{W}$:

find a shortest path $s_w$ and

**if** $s_w \notin \mathcal{P}_w^k$

**then** $\mathcal{P}_w^k = \mathcal{P}_w^k \cup \{s_w\}$ and $F_{s_w}^k = 0$.

**end**

Step 2. (Stopping criterion) If $F^k$ satisfies a stopping criterion, then STOP.

Step 3. (Flow update)

**a.** ($\overline{F}^k$ computation) For each pair $w \in \mathcal{W}$ compute

$$\overline{F}_w^k = \mathrm{Proj}_{X_w^k}(F_w^k - \alpha_k\, C_w(F^k)).$$

**b.** ($\alpha_k$ check)

**while** $\alpha_k > \beta \frac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|}$ **do**

reduce $\alpha_k$:

$$\alpha_k = \min\left\{\xi\,\alpha_k,\ \beta\,\frac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|}\right\},$$

**end**

**c.** (Compute $F^{k+1}$ and re-initialize $\alpha_k$) For each pair $w \in \mathcal{W}$ compute

$$F_w^{k+1} = \mathrm{Proj}_{X_w^k}(F_w^k - \alpha_k\, C_w(\overline{F}^k)),$$

$$\text{set } \alpha_{k+1} = \min\left\{\overline{\alpha}, \beta\frac{\|F^k - \overline{F}^k\|}{\|C(F^k) - C(\overline{F}^k)\|}\right\}.$$

For each pair $w \in \mathcal{W}$ set $\mathcal{P}_w^{k+1} = \mathcal{P}_w^k$, $k = k+1$, and go to step 1.

It is clear that a careful implementation is important for the algorithm to perform well. In particular we need to give a procedure for finding shortest paths, the choice of the initial flow, a stopping criterion and a procedure for performing the projection onto a simplex.

# 4 Implementation of the algorithm

In this section we discuss the main computer implementation issues.

## 4.1 Shortest path subproblem

The proposed algorithm requires iterated solutions of the shortest path problem for each column generation step. Since the shortest paths between all O/D pairs need to be determined simultaneously before performing a column generation step, the Dijkstra's algorithm [8] has been used to find a shortest path tree rooted from each origin, so that the shortest paths are determined for all O/D pairs with the same origin. Moreover, it is not necessary to perform a column generation step in every iteration as we expect the number of paths generated after the first iterations to be very small. Therefore in order to reduce the number of the shortest path computations, we considered a column generation step every $n$ iterations, with $n$ a number to be specified. By performing numerical experiments (see Section 5, Figure 1) we found that the best value of this number varies between 8 and 12, depending on the network being considered.

## 4.2 Finding the initial solution

Usually, in the implementation of the algorithms for the traffic assignment problem, the initial solution is obtained by sending flow on shortest paths with respect to the cost vector corresponding to the zero path flow, referred to as *all-at-once* assignment. In our implementation, we have refined the initial solution as follows: starting with the zero flow we have successively updated the flow and its cost, considering the

origins in sequence. Namely, instead of carrying out the shortest path procedures with respect to all the origins, we have performed that procedure for a single origin and then we have revaluated the flow vector and the corresponding path cost vector. The initial flow thus obtained will be referred to as *once-at-a-time* assignment. This procedure is sketched below:

Set $F = 0$ and evaluate arc costs.

**for** $i = 1$ **to** $r$ **do**

- Build a shortest path tree $\mathcal{T}_i$ rooted in $o_i$. For all $w \in \mathcal{W}$ with origin $o_i$, let $s_w$ be a shortest path in $\mathcal{T}_i$ with respect to $w \in \mathcal{W}$.
- For all $w \in \mathcal{W}$ with origin $o_i$, set $F_{s_w} = d_w$.
- Update arc costs with the current path flow $F$.

**end**

The effect in terms of CPU time of the two different starting points is reported in Table 3, Section 5.

### 4.3 Algorithm for projecting onto a simplex

It is important to remark that all projections on simplices $X_w$ can be performed in a very quick way. According to [22], we consider the following iterative procedure for finding the projection of a vector $z \in \mathbb{R}^n$ onto the simplex $\left\{ x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_i = d \right\}$.

Step 0. Set $k = 0$ and $x_i^0 = z_i + \dfrac{1}{n}\left( d - \sum_{j=1}^n z_j \right)$ for all $i = 1, \ldots, n$.

Step 1. **if** $x^k \geq 0$

**then** STOP

**else** compute $I = \{ i : \ x_i^k > 0 \}$.

**end**

Step 2. For all $i = 1, \ldots, n$ compute

$$x_i^{k+1} = \begin{cases} 0 & \text{if } i \notin I, \\ x_i^k + \dfrac{1}{|I|}\left( d - \sum_{j \in I} x_j^k \right) & \text{if } i \in I, \end{cases}$$

set $k = k + 1$, and go to step 1.

This algorithm performs very simple computations and the projection of $z$ on the simplex $\left\{ x \in \mathbb{R}^n_+ : \sum_{i=1}^n x_i = d \right\}$ is found in at most $n$ iterations.

### 4.4 Stopping criterion

As already pointed out, an efficient stopping criterion is essential when solving the traffic assignment problem. In the literature there is a number of different stopping criteria, most of them based on the difference between two successively computed arc flows vectors, so that the iterative procedure stops when this difference is less than a given accuracy. However, these criteria do not allow us to know what the reached approximation with respect to the equilibrium flow is. In our computational experiments we used a stopping criterion based directly on the Wardrop equilibrium principle: the algorithm is stopped when for each O/D pair the travel times at the paths carrying positive flow approximately equal the travel time of the shortest path. To formalize this fact, we adopted the following error function:

$$\phi(F) = \max_{w \in \mathcal{W}} \left( \frac{1}{d_w} \sum_{p \in \widetilde{\mathcal{P}}_w} F_p \right),$$

where

$$\widetilde{\mathcal{P}}_w = \left\{ p \in \mathcal{P}_w : \ \frac{C_p(F) - C_w^{\min}(F)}{C_w^{\min}(F)} > \delta \right\},$$

$C_w^{\min}(F)$ is the shortest path cost relative to the O/D pair $w$; $\delta > 0$ is a fixed tolerance (e.g., $\delta = 0.01$). This function computes, for each O/D pair, the portion of the demand carried on the paths whose relative difference with respect to the shortest path cost exceeds $\delta$. We remark that if $F^*$ is an equilibrium flow then $\phi(F^*) = 0$ for each tolerance $\delta > 0$. Thus, a very small value of $\phi(F)$ (e.g., less than $10^{-6}$) guarantees that an equilibrium flow is actually achieved. Other choices of error functions, such as the difference of the norm between successive iterates, are less suitable. Indeed, even when the iterates are far from an equilibrium solution, the difference of the norm between the iterates may be small (see Section 5, Table 5). In the next section we present numerical results for some medium and large-scale networks.

## 5 Computational results

This section is devoted to the description of the numerical experiments and the analysis of the results. The proposed algorithm was implemented under MATLAB 7.0.4 and tested on an Intel Pentium 4 at 2.80 GHz, 512MB RAM, running under Windows XP. The numerical tests have been performed on 5 traffic networks: three networks commonly used in literature [1] and two real-life ones [28]. The instances we have considered are described in Table 1.

| instance | nodes | arcs | O/D pairs |
|---|---|---|---|
| Sioux-Falls | 24 | 76 | 528 |
| Arezzo | 213 | 598 | 2,423 |
| Lazio | 306 | 926 | 5,683 |
| Barcelona | 1,020 | 2,522 | 7,922 |
| Winnipeg | 1,052 | 2,836 | 4,345 |

Table 1: test instances.

In [1] the networks are considered with a separable cost function – namely the cost on each arc depends only on the flow on that arc – introduced by the Bureau of Public Roads (BPR):

$$c_a(f) = t_a \left[ 1 + 0.15 \left( \frac{f_a}{K_a} \right)^4 \right],$$

where $c_a(f)$ is the travel cost on arc $a$ at the flow $f$, $t_a$ is the travel cost at zero flow (*free flow travel time*), $f_a$ is the flow on the arc $a$, and $K_a$ is the capacity of the arc $a$. However, to handle real-life instances where there is interaction among traffic on different arcs of the transportation network – as for example two-way streets – the travel cost on each arc is considered to depend upon the flow on other arcs of the network. To deal with asymmetric models, an appropriate cost function is obtained modifying the BPR by adding a term which takes into account the traffic flow in the opposite directions:

$$c_a(f) = t_a \left[ 1 + 0.15 \left( \frac{f_a + 0.5\, f_{\overline{a}}}{2\, K_a} \right)^4 \right],$$

where $\overline{a}$ denotes the opposite arc of $a$. This function leads to an asymmetric equilibrium problem. The same structure of the cost function is also used for the Arezzo and Lazio networks. Detailed data for these networks are given in [28].

In the implementation we set parameters as follows: $\beta = 0.8$, $\xi = 0.9$ and $\bar{\alpha} = 10^6$. We performed the column generation step every 10 iterations and as initial flow we considered the *once-at-a-time* assignment. In order to obtain an accurate solution, we applied the stopping criterion $\phi(F) \leq 10^{-6}$.

We summarize the computational results of our algorithm in Table 2. The first column reports the name of the instance, the second one contains the number of iterations (iter) computed, the third one shows the number of shortest path tree (spt) computations, the fourth and fifth columns give, respectively, the total number of projections (proj) and arc cost function evaluations (cf) executed. The sixth column provides the maximum dimension of each projection computed (dimproj), namely the number of variables in each subproblem, and finally the last column is devoted to the CPU time[1]. It is important to note that the dimension of each computed projection is very small. This allows us to handle equilibrium problems with a great number of O/D pairs.

| instance | iter | spt | proj | cf | dimproj | CPU time |
|---|---|---|---|---|---|---|
| Sioux-Falls | 72 | 264 | 11,693 | 195 | 3 | 1.32 |
| Arezzo | 42 | 612 | 8,553 | 164 | 2 | 7.13 |
| Lazio | 64 | 920 | 47,652 | 243 | 3 | 25.71 |
| Barcelona | 120 | 1,164 | 156,526 | 372 | 3 | 130.14 |
| Winnipeg | 72 | 1,080 | 64,038 | 305 | 2 | 73.70 |

Table 2: computations results.

To show how the performance of the algorithm depends on the number of column generation steps performed, in Figure 1 we report the computational times for the three test scenarios Lazio, Barcelona and Winnipeg networks [1, 28], obtained by changing the number of iterations in which the column generation step is performed.

In Table 3 we summarize the effect of considering two different starting flows, *all-at-once* and *once-at-a-time* assignments, in terms of CPU time.

In [19] the author proposed an extragradient method with a stepsize choice similar to the one described in Section 3, but without re-initialization of the parameter $\alpha$. We have then implemented another version of the algorithm without re-initialization of the parameter $\alpha$. Table 4 shows a comparison between the two versions, with or without $\alpha$ re-initialization.

[1]CPU time (in seconds) is obtained by using the Matlab function `cputime`.

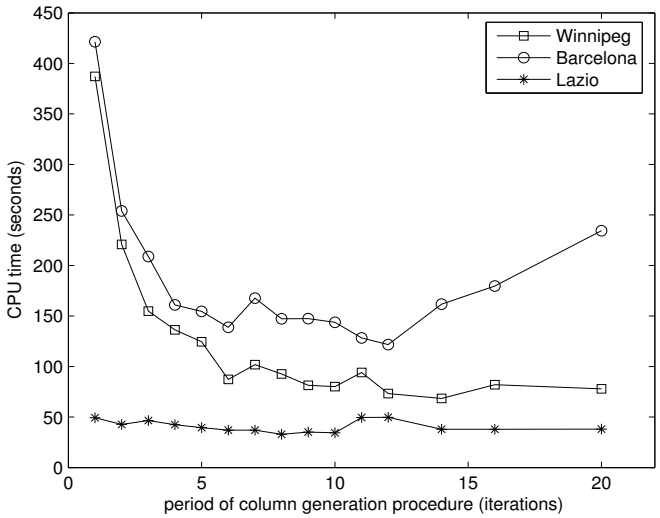


Figure 1: CPU times for different periods of column generation procedure.

| | initial flow | |
|---|---|---|
| | all-at-once | once-at-a-time |
| instance | (CPU time) | (CPU time) |
| Sioux-Falls | 2.00 | 1.32 |
| Arezzo | 7.71 | 7.13 |
| Lazio | 42.59 | 25.71 |
| Barcelona | 991.65 | 130.14 |
| Winnipeg | 785.48 | 73.70 |

Table 3: *all-at-once* vs *once-at-a-time* initial flows.

| instance | $\alpha$ re-initialization with (CPU time) | without (CPU time) |
|---|---|---|
| Sioux-Falls | 1.32 | 53.45 |
| Arezzo | 7.13 | 924.64 |
| Lazio | 25.71 | 2,286.50 |
| Barcelona | 130.14 | 4,600.12 |
| Winnipeg | 73.70 | 1,136.75 |

Table 4: CPU times with or without $\alpha$ re-initialization.

A comparison between the computational times shows that the re-initialization of $\alpha$ improves the overall performance of the algorithm allowing the CPU time to be substantially reduced.

To show the importance of the path-based algorithm in obtaining the desired level of accuracy of the solution, we have also considered a stopping criterion based on arc flow formulation. Given the sequence of arc flows corresponding to the sequence of path flows, we stop the procedure when the average relative error is less than 1%, namely

$$\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A}}\left|\frac{f_a^{k+1}-f_a^k}{f_a^k}\right| < 1\%. \tag{5}$$

Let $f^{\text{path}} = (f_a^{\text{path}})_{a\in\mathcal{A}}$ denote the arc flow vector solution obtained using the path-based stopping criterion $\phi(F) \le 10^{-6}$ and let $f^{\text{arc}} = (f_a^{\text{arc}})_{a\in\mathcal{A}}$ denote the arc flow vector solution obtained using the arc-based stopping criterion (5). In Table 5 we report the average relative error, i.e.

$$\frac{1}{|\mathcal{A}|}\sum_{a\in\mathcal{A},\ f_a^{\text{path}}>0}\left|\frac{f_a^{\text{arc}}-f_a^{\text{path}}}{f_a^{\text{path}}}\right|$$

and the maximum relative error, i.e.

$$\max_{a\in\mathcal{A},\ f_a^{\text{path}}>0}\left|\frac{f_a^{\text{arc}}-f_a^{\text{path}}}{f_a^{\text{path}}}\right|$$

of the solution $f^{\text{arc}}$ with respect to $f^{\text{path}}$.

| instance | average relative error | maximum relative error |
|---|---|---|
| Sioux-Falls | 0.19460 | 0.86343 |
| Arezzo | 0.14268 | 7.36540 |
| Lazio | 0.10208 | 4.19579 |
| Barcelona | 0.24295 | 18.82550 |
| Winnipeg | 0.21228 | 48.42120 |

Table 5: average and maximum relative error of the arc flow obtained with the arc-based stopping criterion (5) with respect to the arc flow obtained with the path-based stopping criterion $\phi(F) \le 10^{-6}$.

These computational results suggest that our path-based algorithm is a valuable alternative to arc-based algorithms in obtaining an accurate solution of the traffic assignment problem.

# 6 Conclusions

We have presented a new iterative method for solving the asymmetric traffic equilibrium problem formulated as a variational inequality with path flow variables. The decomposable structure of the constraint set allows us to use a projection-type method. Our computational experiments show the applicability of the proposed method for solving large-scale problems.

# Acknowledgements

This work has been supported by the National Research Program FIRB/RBNE01WBBBB on Large Scale Nonlinear Optimization. We wish to thank Professor Antonio Pratelli of the Department of Civil Engineering, University of Pisa, for providing us with Arezzo network data and many helpful comments.

# References

[1] H. Bar-Gera, *Transportation Network Test Problems*. `http://www.bgu.ac.il/~bargera/tntp/`.

[2] M. J. Beckmann, C. B. McGuire and C. B. Winsten, *Studies in the Economics of Transportation* , Yale University Press, New Haven, Connecticut, 1956.

[3] D. P. Bertsekas and E. M. Gafni, *Projection Methods for Variational Inequalities with Application to the Traffic Assignment Problem*. Mathematical Programming Study, **17** (1982), 139–159.

[4] A. Chen, D.-H. Lee, and R. Jayakrishnan, *Computational Study of State-of-the-Art Path-Based Traffic Assignment Algorithms*, Mathematics and Computers in Simulation, **59** (2002), 509–518.

[5] A. Chen, D.-H. Lee, and Y. Nie, *A Conjugate Gradient Projection Algorithm for the Traffic Assignment Problem*, Mathematical and Computer Modelling, **37** (2003), 863–878.

[6] S. Dafermos, *Traffic Equilibrium and Variational Inequalities*, Transportation Science, **14** (1980), 42-54.

[7] S. Dafermos and F. Sparrow, *The Traffic Assignment Problem for a General Network*. Journal of Research of the National Bureau of Standards, **73B** (1969), 91–118.

[8] E. W. Dijkstra, *A Note on Two Problems in Connexion with Graphs*, Numerische Mathematik, **1** (1959), 269–271.

[9] F. Facchinei and J. S. Pang, *Finite-Dimensional Variational Inequalities and Complementarity Problems*, Vol. I, Vol. II, Springer Series in Operations Research, Springer-Verlag, New York, 2003.

[10] P. Ferrari, *Road Pricing and Network Equilibrium*, Transportation Research B, **29B** (1995), 357–372.

[11] P. Ferrari, *Optimal Flow Pattern in Road Networks*, in *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Methods*, vol. 58 of *Nonconvex Optim. Appl.*, Kluwer Acad. Publ. Dordrecht, 1995, 101–117.

[12] D. Bernstein and S. A. Gabriel, *The Traffic Equilibrium Problem with Nonadditive Path Costs*, Transportation Science, **31** (1997), 337–348.

[13] D. W. Hearn, S. Lawphongpanich and J. A. Ventura , *Restricted Simplicial Decomposition: Computation and Extensions*, Mathematical Programming Study, **31** (1987), 99–118.

[14] B. J. Kanninen, *Intelligent Transportation Systems: An Economic and Environmental Policy Assessment*. Transportation Research, **30A** (1996), 1-10.

[15] E. N. Khobotov, *Modification of the Extra-Gradient Method for Solving Variational Inequalities and Certain Optimization Problems*, U.S.S.R. Computational Mathematics and Matematical Physics, **27** (1987), 120–127.

[16] G. M. Korpelevich, *The Extragradient Method for Finding Saddle Points and Other Problems*, Matekon, **13** (1977), 35–49.

[17] T. Larsson, J. T. Lundgren, C. Rydergren and M. Patriksson, *Most Likely Traffic Equilibrium Route Flows Analysis and Computation*, in *Equilibrium Problems: Nonsmooth Optimization and Variational Inequality Methods*, vol **58** *Nonconvex Optim. Appl.*, Kluwer, Dordrecht, 2001, 129–159.

[18] T. Leventhal, G. Nemhauser and L. Trotter, *A Column Generation Algorithm For Optimal Traffic Assignment*, Transportation Science, **7** (1973), 168–172.

[19] P. Marcotte, *Application of Khobotov's Algorithm to Variational Inequalities and Network Equilibrium Problems*, INFOR, **29** (1991), 258–270.

[20] P. Marcotte and J. Guélat, *Adaptation of a Modified Newton Method for Solving the Asymmetric Traffic Equilibrium Problem*, Transportation Science, **22** (1988), 112–124.

[21] A. Maugeri, *Convex Programming, Variational Inequalities, and Applications to the Traffic Equilibrium Problem*, Appl. Math. Optim., **16** (1987), 169–185.

[22] C. Michelot, *A Finite Algorithm for Finding the Projection of a Point onto the Canonical Simplex of* $\mathbb{R}^n$, Journal of Optimization Theory and Applications, **50** (1986), 195–200.

[23] A. Nagurney, *Network Economics: a Variational Inequality Approach*, Kluwer, Dordrecht, 1993.

[24] A. Nagurney, *Comparative Tests of Multimodal Traffic Equilibrium Methods*, Transportation Research, **18B** (1984), 469–485.

[25] A. Nagurney and D. Zhang, *Projected Dynamical Systems in the Formulation, Stability Analysis, and Computation of Fixed-Demand Traffic Network Equilibria*, Transportation Science, **31** (1997), 147–158.

[26] A. Nagurney and D. Zhang, *Projected Dynamical Systems and Variational Inequalities with Applications*, Kluwer, Boston, Massachusetts, 1996.

[27] S. Nguyen and C. Dupuis, *An Efficient Method for Computing Traffic Equilibria in Networks with Asymmetric Transportation Costs*, Transportation Science, **18** (1984), 185–202.

[28] M. Passacantando, *Transportation Network Test Problems*, `http://www2.ing.unipi.it/~d9762/`.

[29] M. Patriksson, *The Traffic Assignment Problem: Models and Methods*, VSP, Utrecht, The Netherlands, 1994.

[30] M. Patriksson, *Algorithms for Computing Traffic Equilibria*, Networks and Spatial Economics, **4** (2004), 23–38.

[31] M. J. Smith, *The Existence, Uniqueness and Stability of Traffic Equilibria*, Transportation Research, **13B** (1979), 295-304.

[32] M. V. Solodov and B. F. Svaiter, *A New Projection Method for Variational Inequality Problems*, SIAM Journal of Control and Optimization, **37** (1999), 765–776.

[33] Y. Wang, N. H. Xiu and C. Y. Wang, *A New Version of Extragradient Method for Variational Inequality Problems*, Computers & Mathematics with Applications, **42** (2001), 969–979.

[34] Y. J. Wang, N. H. Xiu and C. Y. Wang, *Unified Framework of Extragradient-Type Methods for Pseudomonotone Variational Inequalities*, Journal of Optimization Theory and Applications, **111** (2001), 641–656.

[35] J. G. Wardrop, *Some Theoretical Aspects of Road Traffic Research*, Proceedings of the Institute of Civil Engineers, **1** (1952), 325-378.