An efficient method to compute traffic assignment problems with elastic demands

F. Babonneau* J.-P. Vial*

Abstract

The traffic assignment problem with elastic demands can be formulated as an optimization problem, whose objective is sum of a congestion function and a disutility function. We propose to use a variant of the Analytic Center Cutting Plane Method to solve this problem. We test the method on instances with different congestion functions (linear with capacities on the arc and BPR) and different demand functions (constant elasticity and linear). The results of the numerical experiments show that it is possible to solve large instances with high accuracy.

Keywords. Traffic assignment problem, Elastic demand, ACCPM.

Introduction

The traffic assignment problem, in short TAP, consists in determining which routes to assign to the drivers who travel on a transportation network from some origins and some destinations. Wardrop [26] enunciated two broad principles for determining the assignment. According to the first principle, the assignment is an equilibrium state in which no user can reduce his travel time by using an alternative route. The second principle states that the assignment should minimize the total travel times of all users. The two principles lead to different assignments. In the first case the assignment is named a *user equilibrium*, and, in the second case, one talks of *system optimum*. An essential point is that in both cases, the assignment can be characterized as an optimal solution of a convex multicommodity flow problem.

In most formulations, the number of drivers who want to travel from a specific origin node to a specific destination node is fixed. In economic terms, the demand associated to an origin/destination pair is inelastic. A more realistic stand considers that the demands may be adversely affected by the travel times. The user equilibrium can be extended to account for elastic demands. In particular, it has been shown [6, 12] that the user equilibrium with elastic demands can be computed as the solution of an optimization problem, whose objective is sum of a congestion function and a disutility demand function. A similar property holds for the system optimum.

*HEC/Logilab, University of Geneva, 40 Bd du Pont d'Arve, CH-1211 Geneva 4, Switzerland

The traffic assignment problem with elastic demands is often studied in the toll pricing framework with elastic demands. As noted earlier the user equilibrium and the system optimum are not equivalent: the user equilibrium is sub-optimal from a social point of view. By adding suitable tolls on the arcs, it is possible to force the user equilibrium to achieve a social optimum. The solution methods for the toll pricing TAP use as a subproblem the TAP with elastic demands.

The elastic TAP is well-studied in the literature [25], at least from a theoretical point of view, but few papers discuss solution methods and their performance. We briefly review two approaches. We first mention the methodology proposed in [12, 13]. The elastic demand TAP is transformed as an equivalent inelastic TAP in an expanded network. A new arc is added in the graph for each commodity. The cost associated with that arc is the disutility of the demand; it plays the same role as a congestion function. This elegant transformation introduces as many arcs as the number of commodity, a very large number in many applications. As a result the equivalent problem is often impractical. The alternative for solving elastic TAP is to use variants of the Frank-Wolfe method [11], either by linearization of two components of the objective function or, as in Evans algorithm [9], a linearization of the congestion function alone and explicit computation of the demand associated with the particular linearization. The numerical studies found in the literature [13, 10, 16, 21] mostly apply to small-size problems. This is in sharp contrast with the case of inelastic demands for which very efficient methods have been proposed to solve extremely large problem instances [3, 7, 14]. More recently, huge instances have been solved [1, 2] using a variant of the Analytic Center Cutting Plane Method (ACCPM) [15]. The purpose of this paper is to use this methodology to compute traffic assignments with elastic demands.

The solution method is based on Lagrangian relaxation. As noted in [2], the Lagrangian dual objective function of TAP has two main components: a piece-wise linear one and one that is the negative Fenchel conjugate of the congestion function. This type of convex non-differentiable problem is well-solved by ACCPM [14]. The key point in this method is that a linear approximation of the epigraph of the two functions can be obtained using subgradients computed at privileged points. These linear approximations, or cutting planes, define, in the epigraph space, a so-called localization set that contains the set of optimal solution. ACCPM improves the quality of the approximation by choosing iteratively the linearization points as the analytic center of the localization set. This choice ensures pseudo-polynomial convergence and excellent performance in practice. This very general method ignores the fact that the second component of the objective of the Lagrangian dual problem, namely the negative of the Fenchel conjugate of the congestion function, is smooth and that its derivatives can be computed explicitly. This knowledge can be exploited to define a new localization set as the combination of a nonlinear set —the one that pertains to the conjugate of the congestion function— and a linear approximation of the other epigraph. The method has been introduced in [2] and has made it possible to solve huge instances. In the present paper we use the same methodology as in [2] and we treat the elastic demand in a way similar to [9].

In our numereical experiments, we consider the most popular congestion functions used in transportation and two different demand disutility functions. The congestion function is either linear with a capacity constraint or the classical BPR (Bureau of Public

Roads) function. The demand function for each origin-destination pair is either a constant-elasticity one, or a linear decreasing one. The new approach has been tested on standard realistic transportation problems that can be found in the open literature. We report results on large instances which we solve with high accuracy.

The paper is organized as follows. In Section 1, we give the formulation of the traffic assignment problem with elastic demands and in Section 2 we present the associated Lagrangian relaxation. Section 3 and Section 4 sketch the solution method and discuss the particular settings of the method parameters. The last section is devoted to the numerical results.

1 Traffic assignment problem

Let $\mathcal{G}(\mathcal{N},\mathcal{A})$ be an oriented graph, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. The graph represents a network on which drivers, or commodities, must be shipped from specific origins to specific destinations. We denote $\mathcal{K}$ the set of commodities. In the traffic assignment problem (TAP), each commodity is characterized by an unique pair of origin and destination nodes. Let N be the node-arc incidence matrix of $\mathcal{G}$ and x represent the flow vector. We define the feasible set of flows as

$$\mathcal{X} = \{x \geq 0 \mid Nx^{\kappa} = d^{\kappa},\ \kappa \in \mathcal{K}\}. \tag{1}$$

The vector d^{κ} has only two non-zero components : $-\delta_{\kappa}$ at the origin and δ_{κ} at the destination of the commodity. In that formulation, δ_{κ} is the demand for commodity κ. We denote δ the vector of all demands.

Given a feasible flow $x \in \mathcal{X}$, we can evaluate for each driver the travel time spent for covering the distance from its origin to its destination. This travel time is the sum of the travel times of the used arcs. We assume that the arc travel time, denoted tt_a, is a positive, convex and non-decreasing function of the total flow on the arc a, denoted y_a. The total travel time is then separable into arc travel times. Let $\mathcal{R}_{\kappa}$ be the set of all possible routes from the origin of the driver κ to its destination. We define the travel time of a route $r \in \mathcal{R}_{\kappa}$ such that

$$\lambda_{\kappa}^{r} = \sum_{a \in r} tt_a(y_a).$$

The standard assumption [17] with elastic demands is as follows.

Assumption 1 *The demand δ_{κ} for commodity κ only depends on the shortest travel time from the supply node to the demand node. The demand function $\delta_{\kappa}(\lambda)$, where λ is the travel time along the shortest path, is a function from $\mathbb{R}^+$ to $\mathbb{R}^+$. It is continuously differentiable, non-negative, upper bounded, and strictly decreasing.*

Assumption 1 implies that the total demand function is separable into commodity demand functions. The other implication is that the inverse function of δ_{κ} exists. We denote it $\lambda_{\kappa}(s) = \delta_{\kappa}^{-1}(s)$, where s is a demand.

In the subsequent developments, we concentrate on the user equilibrium. The case of the system equilibrium is very similar, but the discussion is omitted. According to the user equilibrium principle, the drivers select their routes independently and the travel times of all used routes are less or equal than those which would be experienced by a single driver on any unused route. This condition can be written as

$$x_\kappa^r > 0 \quad \Rightarrow \quad \lambda_\kappa^r = \min_{p \in \mathcal{R}_\kappa} \lambda_\kappa^p, \quad r \in \mathcal{R}_\kappa, \quad \kappa \in \mathcal{K}, \tag{2a}$$

$$x_\kappa^r = 0 \quad \Rightarrow \quad \lambda_\kappa^r \geq \min_{p \in \mathcal{R}_\kappa} \lambda_\kappa^p, \quad r \in \mathcal{R}_\kappa, \quad \kappa \in \mathcal{K}. \tag{2b}$$

When the travel time and demand functions are separable and integrable, the equilibrium conditions (2) are solution of the optimization problem (see [25]):

$$\min_{x,y,\delta} \quad \sum_{a \in \mathcal{A}} \int_0^{y_a} tt_a(s)ds - \sum_{\kappa \in \mathcal{K}} \int_0^{\delta_\kappa} \lambda_\kappa(s)ds \tag{3a}$$

$$y = \sum_{\kappa \in \mathcal{K}} x^\kappa, \tag{3b}$$

$$x \in \mathcal{X}(\delta). \tag{3c}$$

Here, $\mathcal{X}(\delta)$ denotes the set of feasible flow in (1).

2 Lagrangian relaxation

For the sake of simpler notations, we define the congestion function

$$g(y) = \sum_{a \in \mathcal{A}} g_a(y_a) = \sum_{a \in \mathcal{A}} \int_0^{y_a} tt_a(s)ds,$$

and the disutility demand function

$$h(\delta) = \sum_{\kappa \in \mathcal{K}} h_\kappa(\delta_\kappa) = -\sum_{\kappa \in \mathcal{K}} \int_0^{\delta_\kappa} \lambda_\kappa(s)ds.$$

We also write the constraint $y = \sum_{\kappa \in \mathcal{K}} x^\kappa$ in this abstract form $y = Mx$. Problem (3) is now written as

$$\min \quad g(y) + h(\delta) \tag{4a}$$

$$Mx = y, \tag{4b}$$

$$x \in \mathcal{X}(\delta), \tag{4c}$$

where M is a matrix collecting the flows on the arcs. If we associate dual variables u with (4b) and relax this constraint, we obtain the so-called Lagrangian dual problem

$$\max_u f(u), \tag{5}$$

where f is defined by

$$f(u) = \min_{y,x,\delta} \{g(y) + h(\delta) + \langle u, Mx - y \rangle \mid x \in \mathcal{X}(\delta)\}. \tag{6}$$

By convexity, the Lagrangian dual problem has the same optimal value as (4).

A quick inspection shows that (6) is separable in the variables y and x. Thus, (6) can be written as

$$f(u) = f_1(u) + f_2(u), \tag{7}$$

where

$$f_1(u) = \min_{x,\delta}\{h(\delta) + \langle M^T u, x\rangle \mid x \in \mathcal{X}(\delta)\},$$

and

$$f_2(u) = \min_{y}\{g(y) - \langle u, y\rangle\}.$$

Recall that the formulation (4) hides the fact that the x and δ variables have different components for each commodities. These variables are linked by (4b), but relaxing this constraint has the effect that $f_1(u)$ and $f_2(u)$ are separable in the commodities. For the sake of a simple presentation, we shall assume that Problem (5) has only one commodity. In this case $\delta \in \mathbb{R}_+$. In our problem of interest, x is the vector of arc flows that ships δ units from an origin node to a destination node. The following assumption is thus verified.

Assumption 2 *$x \in \mathcal{X}(\delta)$ if and only if $\frac{1}{\delta}x \in \mathcal{X}(1)$.*

Computing $f_1(u)$ is equivalent to the two-stage minimization

$$f_1(u) = \min_{\delta}\left\{h(\delta) + \min_{x}\{\langle M^T u, x\rangle \mid x \in \mathcal{X}(\delta)\}\right\}.$$

In view of Assumption 2

$$\min_{x}\{\langle M^T u, x\rangle \mid x \in \mathcal{X}(\delta)\} = \delta \min_{x}\{\langle M^T u, x\rangle \mid x \in \mathcal{X}(1)\}.$$

For convenience, we denote

$$\xi_u = \arg\min_{x}\{\langle M^T u, x\rangle \mid x \in \mathcal{X}(1)\}.$$

Thus

$$f_1(u) = \min_{\delta}\{h(\delta) + \langle M^T u, \xi_u\rangle \delta\}.$$

In the considered applications, the function h is convex and differentiable. Moreover, $h'(\delta) = -\lambda(\delta)$ is strictly decreasing. Therefore, the minimum is achieved at

$$\delta_u = -(h')^{-1}(\langle M^T u, \xi_u\rangle) = \arg\min_{\delta}\{h(\delta) + \langle M^T u, \xi_u\rangle \delta\}. \tag{8}$$

Note that $(h')^{-1}(l) = -\delta(l)$, where l is a travel time.

To implement a cutting plane, we need to be able to compute anti-subgradients of f_1. Let u and u' be two different points and define the quantities

$$\begin{aligned}
f_1(u') &= \min_{x,\delta}\{h(\delta) + \langle M^T u', x\rangle \mid x \in \mathcal{X}(\delta)\}\\
&= h(\delta_{u'}) + \langle M^T u', \xi_{u'}\rangle \delta_{u'}\\
&\leq h(\delta_u) + \langle M^T u', \xi_u\rangle \delta_u\\
&= h(\delta_u) + \langle M^T u', \xi_u\rangle \delta_u + \langle (M\xi_u)\delta_u, u' - u\rangle \delta_u\\
&= f_1(u) + \langle a, u' - u\rangle,
\end{aligned}$$

with $\gamma = (M\xi_u)\delta_u$. This proves that $f_1(u)$ is convex and that $a \in -\partial(-f_1(u))$ is an anti-subgradient of f_1.

Akin, the function $f_2(u)$ is the point-wise minimum of a collection of affine functions of u. It is thus concave and one may construct an inequality. Actually, we can get more. From the definition, we observe that $f_2(u)$ is the opposite of the Fenchel conjugate $g_*(u)$ of g. In the cases under study, $g_*(u)$ can be given in closed form and it also appears to be twice continuously differentiable. We certainly want to exploit this property when it is verified, and devise more efficient algorithms to solve the Lagrangian dual problem.

In view of the definition of the congestion function g, the right derivative $g'_+(y)$ of g at $y = 0$ is well-defined. From the first optimality conditions of (4) and since g is monotone increasing, the constraint

$$u \geq g'_+(0) = u_l, \tag{9}$$

is always met at the optimum. It is nevertheless convenient to introduce this redundant constraint in the formulation of problem (5).

3 ACCPM (with a proximal term)

We aim to solve (5) with a version of ACCPM which exploits the fact that the smooth component f_2 of the objective function is convave with explicit first and second derivatives. We introduce the constraint $u \geq u_l$ into (5) and work with the canonical version

$$\max\{f(u) = f_1(u) + f_2(u) \mid u \geq u_l\}, \tag{10}$$

in which $u \in \mathbb{R}^m$, $f_1 : \mathbb{R}^m \to \mathbb{R}$ is a concave function and $f_2 : \mathbb{R}^m \to \mathbb{R}$ is a concave, twice continuously differentiable function. Information on these functions is delivered by *oracles.*

Definition 1 *A first order oracle for the concave function $h : \mathbb{R}^m \to \mathbb{R}$ is a black-box procedure that returns a support to h at the query point $\bar{u}$:*

$$a^T(u - \bar{u}) + h(\bar{u}) \geq h(u), \quad \forall u \in \operatorname{dom} h, \tag{11}$$

where the vector $a \in -\partial(\text{-}h(\bar{u})) \subset \mathbb{R}^m$ is an element of the anti-subgradient set.

Definition 2 *A second-order oracle for the concave function $h : \mathbb{R}^m \to \mathbb{R}$ is a black box procedure with the following property. When queried at $\bar{u}$, the oracle returns the function value and the first and second derivatives of $h(u)$ at $u = \bar{u}$.*

The first order oracle f_1 has been presented in the previous section. We shall compute later the explicit form of f_2 for our applications. For the time being, we assume that f_2 is accessed through a second-order oracle.

3.1 The algorithm

The hypograph set of the function f is the set defined by $\{(z+\zeta, u) \mid z \leq f_1(u),\ \zeta \leq f_2(u)\}$. Optimality cuts (11) provide an outer polyhedral approximation of the hypograph set of the concave function f_1. Suppose that a certain number of query points u^k, $k = 1, \dots, K$, have been generated. The associated anti-subgradients $a^k \in -\partial(\text{-}f_1(u^k))$ are collected in a matrix A. We further set $\gamma_k = f_1(u^k) - \langle a^k, u^k \rangle$. The polyhedral approximation of the hypograph set of f_1 is $\gamma \geq ze - A^T u$, where e is the all-ones vector of appropriate dimension. Finally, let $\underline{\theta}$ be the best recorded value: $\underline{\theta} = \max_{k \leq K}\{f_1(u^k) + f_2(u^k)\}$.

In view of the above definitions, we can define the so-called *localization set*, which is a subset of the hypograph of f

$$\mathcal{F}_{\underline{\theta}} = \{(u, z, \zeta) \mid -A^T u + ze \leq \gamma,\ \zeta \leq f_2(u),\ z + \zeta \geq \underline{\theta},\ u \geq u_l\}. \tag{12}$$

Clearly, the set contains all optimal pairs $(u^*, f(u^*))$. Thus, the search for a solution should be confined to the localization set. The selection of a point within the localization set is a crucial ingredient of a cutting plane method. Leaving this critical issue aside for a while, we sketch the basic step, or *outer iteration*, of a generic cutting plane method.

Algorithm 1: Outer iteration of constrained ACCPM

1. Select a query point in the localization set.
2. Send the query point to the first order oracle and get back an optimality cut to f_1.
3. Send the query point to the second order oracle to compute the objective function f_2.
4. Update the lower and upper bounds and the localization set.
5. Test termination.

3.2 Proximal analytic centers

In the proposed version of ACCPM, the query point is an approximate proximal analytic center of the localization set defined as the intersection of cutting planes and a fixed cutting surface. The proximal analytic center is defined as the unique minimizer of a logarithmic barrier for the localization set, augmented with a proximal term. The analytic center is the u component of the solution (u, z, ζ) to the minimization problem

$$\min \quad F(u, z, \zeta) = \frac{\rho}{2}||u - \underline{u}||^2 - \sum_{i=0}^{K} \log s_i - \log \sigma - \sum_{i=1}^{m} \log(u_i - u_{li}) \tag{13a}$$

$$s_0 = z + \zeta - \underline{\theta} \geq 0, \tag{13b}$$

$$s_i = \gamma_i - z + (a^i)^T u \geq 0, \quad i = \{1, \dots, K\}, \tag{13c}$$

$$\sigma = f_2(u) - \zeta \geq 0. \tag{13d}$$

Note that $F(u, z, \zeta)$ is defined on the interior of the localization set $\mathcal{F}_{\underline{\theta}}$. The proximal reference point $\underline{u}$ and the proximal coefficient ρ are arbitrary. In practice, $\underline{u}$ is chosen to be the query point u^k that achieves the best recorded value $\underline{\theta}$, i.e., $\underline{u} = \arg\max_{k \leq K}\{f_1(u^k) + f_2(u^k)\}$.

Remark 1 *It is easy to show that $F(u, z, \zeta)$ achieves its minimum value when the localization set has a non-empty interior. Moreover, this minimum is unique.*

The algorithm that computes the analytic center is a damped Newton method applied to the first order optimality conditions of Problem (13). See [2] for more details. Each iteration of the damped Newton method is named an *inner iteration.*

3.3 Upper and lower bounds

By duality, any feasible solution of (10) provides a lower bound for the original problem (3). Taking the values returned by the two oracles at the successive query points, we obtain the lower bound

$$\underline{\theta} = \max_{k \leq K}\{f_1(u^k) + f_2(u^k)\}. \tag{14}$$

An upper bound $\bar{\theta}$ can be obtained from information collected in the computation of the analytic center. As descibed in [1], ACCPM provides at each iteration a convex combination of flow vectors in $\mathcal{X}$: it also belongs to $\mathcal{X}$. This convex combination is directly used in the primal objective to compute the upper bound $\bar{\theta}$.

4 Implementation issues

In this section, we review the main items in the implementation of our solution method.

4.1 First order oracle

The first order oracle is a two-stage optimization process. In the first stage, the first order oracle consists of $|\mathcal{K}|$ shortest path computations, using Dijkstra's algorithm [8]. This algorithm computes shortest paths from a single node to all other nodes in a directed graph. To compute the shortest paths for all commodities, we partition the commodities according to the origin node of the demand. For each origin node, we compute in the same pass of Dijkstra's algorithm the shortest paths for all the commodities having their origin in that node. The number of solves is thus at most $|\mathcal{N}|$, the cardinality of the node set. For large graphs, most of the computational time is devoted to data handling. To speed-up computation, the algorithm is implemented with binary heap structures. This implementation is efficient enough, but probably not on par with the state-of-the-art. A better implementation could improve the performance of the overall algorithm, but only marginally.

In the second stage, the first order oracle uses the length of the shortest path of each commodity to compute its demand to be assigned on its shortest path using (8).

4.2 Parameter settings in ACCPM

Few parameters have to be set in ACCPM. The important ones are the coefficient of the proximal term and the proximal reference point; and the weight on the logarithmic barrier on the floor cut.

Proximal reference point and proximal coefficient The initial proximal reference point is the first query point. Thereafter, the proximal reference point is updated to the current query point whenever the oracle returns an objective function value that improves upon the best lower bound. The value for the proximal coefficient ρ is 10^{-5} and it is not updated during the process.

Weight on floor cut The localization set is bounded below by the special constraint $z + \zeta \geq \underline{\theta}$ in (12). We name it the *floor cut*. It is easily checked that the floor cut makes a negative angle with the cutting planes. When the number of cutting planes increases, their total weight dominates the weight of the floor cut in (13). Thus, the floor cut tends to become active at the analytic center, with the possible effect of slowing the global convergence. To counteract this negative effect, we put a weight to the floor cut that equals the total number of generated cuts.

4.3 Termination criterion

The standard termination criterion is a small enough relative optimality gap:

$$(\bar{\theta} - \underline{\theta})/\max(\underline{\theta}, 1) \leq \epsilon, \tag{15}$$

where $\underline{\theta}$ is the best lower bound computed with (14) and $\bar{\theta}$ is the best upper bound. In our experiments we use $\epsilon = 10^{-5}$.

5 Numerical experiments

The main goal of our empirical study is to test the efficiency of our method on traffic assignment problems with elastic demands. We solve large size instances using two congestion functions and two demand disutility functions.

5.1 Travel time functions

In this subsection, we introduce the two travel time functions mainly used in traffic. We also give the associated congestion functions.

In traffic assignment problem, the literature essentially deals with the *BPR* (Bureau of Public Roads) function [3, 4, 7, 19]. The *BPR* congestion function is

$$g_a(y_a) = r_a y_a \left(1 + \frac{p}{q+1}(\frac{y_a}{c_a})^q\right), \quad \text{with } y_a \geq 0. \tag{16}$$

The associated travel time is

$$tt_a(y_a) = r_a + \frac{r_a p}{c_a^q} y_a^q, \quad \text{with } y_a \geq 0. \tag{17}$$

In general, the parameter p is very small and $q > 1$ does not exceed 5. When the flow y_a is less than c_a, the second term under the parenthesis in (16) is negligible. Thus $g_a(y_a) \approx r_a y_a$. The parameter r_a is called *free-flow travel time*; it can be interpreted as a fixed travel time on a congestion-free arc. For larger values of y_a the nonlinear contribution to congestion increases. The threshold value c_a for the flow y_a is usually named the *practical capacity* of the arc, beyond which congestion becomes effective. In some applications, the parameters p and q are arc-dependent.

The second main function used in traffic modeling is the linear congestion function with a capacity on the total flow [23, 1, 20]. Here, the travel time is given by

$$tt_a(y_a) = \begin{cases} t_a, & \text{if } y_a \leq c_a, \\ +\infty, & \text{otherwise.} \end{cases}$$

where $t_a \geq 0$ is a constant and c_a is the capacity on the total flow. The associated congestion function is the linear function

$$g_a(y_a) = \begin{cases} t_a y_a, & \text{if } y_a \leq c_a, \\ +\infty, & \text{otherwise.} \end{cases}$$

5.2 Demand disutility functions

In this subsection, we present the two demand functions used to experiment our method. Table 1 displays the generic demand functions and the associated disutility functions. The parameters α and β actually depend on the commodity $\kappa \in \mathcal{K}$. The first demand function,

Function	Demand $\delta(\lambda)$	Disutility $h(\delta)$
Constant elasticity	$\beta e^{-\alpha\lambda}$	$\frac{\delta}{\alpha}[\log(\frac{\delta}{\beta}) - 1]$
Linear	$\max(0, -\alpha\lambda + \beta)$	$\frac{1}{2\alpha}\delta^2 - \frac{\beta}{\alpha}\delta$

Table 1: Demand and disutility functions.

used in [21], has a constant elasticity $-\alpha$; the second one, used in [16, 10, 21], is a linear and decreasing demand function. The linear demand function is actually piece-wise linear. It is not strictly decreasing and does not match Assumption 1. This could be a problem in equation (8) because the inverse function of $h'(\delta) = \lambda(\delta)$ is not formally defined for travel time larger than β/α. Actually, this is not an issue, because for all travel times $\langle M^T u, \xi_u \rangle \geq \beta/\alpha$, we can take $\delta_u = 0$.

The data on which we perform our numerical experiments deal with fixed demands. We set β_κ to be equal to this fixed demand. Indeed, both demand functions tend to β when α tends to zero, that is when the demand becomes inelastic. In the experiments we use different values of α, to see how sensitive is the method to elasticity.

5.3 Test problems

In the numerical experiments, we use a collection of problems composed of three realistic transportation problems used in [1, 3, 7, 19]. The data are adapted for the BPR function. They include *free-flow travel time*, *practical capacity* and the tuning parameters p and q. Since this set of problems is originally devoted for the traffic assignment with fixed demands, the data also provide demand which are used to calibrate the disutility demand functions. These problems, can be downloaded from `http://www.bgu.ac.il/~bargera/tntp/`.

To solve these problems with the linear congestion function, we use the available *practical capacity* as capacity on the total flow. The demand used to calibrate the disutility demand function are those used in [1, 20]. These demands are made feasible with respect to the practical capacity in the fixed demand model.

Problem ID	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{K}\rvert$	z^*_{Linear}	z^*_{BPR}
`Sioux-Falls`	24	76	528	3.20184×10^5	4.23133×10^6
`Winnipeg`	1067	2975	4345	2.94065×10^7	8.25672×10^5
`Barcelona`	1020	2522	7922	3.89400×10^7	1.23277×10^6

Table 2: Test problems.

Table 2 displays problem data. For each problem instance, we give the number of nodes $|\mathcal{N}|$, the number of arcs $|\mathcal{A}|$ and the number of commodities $|\mathcal{K}|$. In the last two columns, we give the optimal values of the BPR and the linear functions for the inelastic traffic problem, z^*_{BPR} and z^*_{Linear}, respectively.

5.4 Numerical results

In this subsection, we carry experiments using the two congestion functions (linear and BPR) and the two demand functions (constant-elasticity and linear). For each possible association of congestion and demand functions, we test different values for the parameter α. All problems are solved with a 10^{-5} relative optimality gap. Since the congestion function and the demand function are from different natures, they may have different order of magnitude. To make the two components comparable we weigh the demand function with a weight evaluated in a preprocessing phase.

Table 3 reports results using the BPR function for Sioux-Falls, Winnipeg and Barcelona, respectively. Similarly, Table 4 reports results using the linear congestion function with fixed capacities on the arcs. Each of these tables is divided in two parts. The upper part contains the results with the constant-elasticity demand function while the lower part gives the results using the linear demand function.

For all results, the tables give the parameter α, the objective value of the congestion part, denoted *Flow obj*, the objective value of the demand disutility part, denoted *Demand obj*, the weight used for the demand function, denoted *Weight*, the number of outer iterations, denoted *Outer*, the number of Newton's iteration, or inner iterations, denoted

Inner, the computational time in seconds *CPU* and the percentage of CPU time denoted *%Or* spent to compute the shortest path problems.

The ACCPM code we use has been developed in Matlab, while the shortest path algorithm is written in C. The tests were performed on a PC (Pentium IV, 2.8 GHz, 2 Gb of RAM) under Linux operating system.

We observe first that all problems with all configurations are solved with the required accuracy with CPU times less 145 seconds. As noted in [2], problems with a linear congestion function and a fixed arc capacity are more difficult than those with the BPR nonlinear congestion function, at least for the two larger problems Winnipeg and Barcelona. When α is very small, we retrieve the solution computed in [1] and [2]. In those papers, the performance (CPU, outer iterations) are superior because they use an active set strategy and a partial linearization of the BPR function. In the present study those features have been turned down. The last columns "%Or" and "Inner" reveal that one third of the computing time is devoted to solving shortest path problems (oracle), and that the average number of Newton steps to find an approximate analytic center is less than 2.25 for BPR and 2.86 for the linear congestion.

Our final remark is on the impact of the elasticity factor. The figures show that the problems are easier for an absolute elasticity 10^{-1} for the BPR congestion function and 10^{-4} for the linear congestion function. For lower and higher values, the number of outer iterations and the CPU are higher. We have no explanation for this behavior.

6 Conclusion

In this paper, we proposed a new method, i.e., ACCPM, for solving the traffic assignment problem with elastic demands. This method already has proved its efficiency on the fixed demand TAP [1, 2]. The numerical experiments are performed on relatively large instances of TAP elastic demand show that ACCPM is also efficient for that class of problems.

α	Flow obj.	Demand obj.	Weight	Outer	Inner	CPU	%Or
Sioux-Falls							
	Constant-elasticity demand function						
10^{-10}	4.23134×10^{6}	-3.60600×10^{15}	10^{-9}	58	139	1.5	32
10^{-6}	4.23114×10^{6}	-3.60600×10^{11}	10^{-5}	56	134	1.5	34
10^{-4}	4.20992×10^{6}	-3.60600×10^{9}	10^{-3}	57	165	1.6	29
10^{-1}	1.11372×10^{6}	-2.71966×10^{6}	1	21	69	0.6	31
1	1.15165×10^{4}	-1.56506×10^{4}	1	24	49	0.6	33
3	9.34039×10^{1}	-1.08521×10^{2}	1	33	67	0.8	32
	Linear demand function						
10^{-10}	4.23134×10^{6}	-2.51030×10^{18}	10^{-12}	57	136	1.5	36
10^{-6}	4.23114×10^{6}	-2.51021×10^{14}	10^{-8}	56	136	1.4	34
10^{-4}	4.21186×10^{6}	-2.50132×10^{12}	10^{-6}	57	137	1.5	34
10^{-1}	4.21946×10^{5}	-1.98689×10^{8}	10^{-3}	12	45	0.3	31
1	1.03892×10^{3}	-8.73232×10^{3}	1	24	49	0.5	36
3	0	0	1	37	75	0.9	35
Winnipeg							
	Constant-elasticity demand function						
10^{-10}	8.25672×10^{5}	-6.47750×10^{14}	10^{-9}	161	355	42.2	33
10^{-6}	8.25658×10^{5}	-6.47751×10^{10}	10^{-5}	163	359	45.5	35
10^{-4}	8.24186×10^{5}	-6.47814×10^{8}	10^{-3}	111	271	25.7	39
10^{-1}	2.29486×10^{5}	-4.64149×10^{5}	1	77	165	14.2	45
1	7.88694×10^{2}	-1.00126×10^{3}	1	137	278	33.7	38
3	2.09473	-2.38610	1	163	330	44.8	36
	Linear demand function						
10^{-10}	8.25672×10^{5}	-1.09019×10^{16}	10^{-10}	158	349	42.1	35
10^{-6}	8.25655×10^{5}	-1.09016×10^{12}	10^{-6}	157	347	41.7	35
10^{-4}	8.23966×10^{5}	-1.08730×10^{10}	10^{-4}	148	329	37.9	36
10^{-1}	1.18043×10^{5}	-1.51622×10^{6}	10^{-1}	66	136	9.8	48
1	4.47438×10^{1}	-8.14318×10^{1}	1	144	292	35.8	37
3	0	0	1	161	326	43.2	35
Barcelona							
	Constant-elasticity demand function						
10^{-10}	1.23278×10^{6}	-1.84677×10^{15}	10^{-9}	133	302	35.6	38
10^{-6}	1.23277×10^{6}	-1.84677×10^{11}	10^{-5}	132	300	34.1	36
10^{-4}	1.23165×10^{6}	-1.84684×10^{9}	10^{-3}	118	291	30.0	34
10^{-1}	5.84859×10^{5}	-1.62221×10^{6}	1	99	214	21.2	39
1	1.04802×10^{4}	-1.41871×10^{4}	1	144	293	39.7	37
3	9.71231×10^{1}	-1.15514×10^{2}	1	174	353	56.3	36
	Linear demand function						
10^{-10}	1.23278×10^{6}	-1.12802×10^{17}	10^{-11}	149	334	42.6	36
10^{-6}	1.23277×10^{6}	-1.12801×10^{13}	10^{-7}	149	334	42.4	36
10^{-4}	1.23152×10^{6}	-1.12684×10^{11}	10^{-5}	146	328	41.0	36
10^{-1}	4.82731×10^{5}	-4.61275×10^{7}	10^{-2}	37	273	10.7	24
1	3.43352×10^{3}	-2.34378×10^{4}	1	130	265	33.3	38
3	0	0	1	185	375	61.6	34

Table 3: Results with the BPR congestion function

α	Flow obj.	Demand obj.	Weight	Outer	Inner	CPU	%Or
Sioux-Falls							
	Constant-elasticity demand function						
10^{-10}	3.20184×10^{5}	-3.60600×10^{14}	10^{-9}	35	113	0.9	30
10^{-6}	3.20180×10^{5}	-3.60600×10^{10}	10^{-5}	35	114	0.9	28
10^{-4}	3.19818×10^{5}	-3.60606×10^{8}	10^{-3}	30	153	0.9	23
10^{-1}	1.17741×10^{5}	-2.83284×10^{5}	1	21	44	0.5	33
1	1.17533×10^{3}	-1.59571×10^{3}	1	32	65	0.7	32
3	9.67088	-1.12348×10^{1}	1	41	83	0.9	34
	Linear demand function						
10^{-10}	3.20184×10^{5}	-2.51030×10^{16}	10^{-11}	34	110	0.9	32
10^{-6}	3.20180×10^{5}	-2.51026×10^{12}	10^{-7}	34	111	0.8	29
10^{-4}	3.19804×10^{5}	-2.50654×10^{10}	10^{-5}	34	112	0.9	33
10^{-1}	1.05194×10^{5}	-7.03503×10^{6}	10^{-1}	16	34	0.3	40
1	5.68039×10^{2}	-3.19020×10^{3}	1	29	59	0.6	38
3	0	0	1	41	83	0.9	38
Winnipeg							
	Constant-elasticity demand function						
10^{-10}	2.94064×10^{7}	-2.39907×10^{14}	10^{-7}	277	899	144.6	19
10^{-6}	2.93679×10^{7}	-2.39966×10^{10}	10^{-3}	265	914	142.0	19
10^{-4}	2.59880×10^{7}	-2.43696×10^{8}	10^{-1}	113	699	42.4	18
10^{-1}	1.52477×10^{-4}	-1.60721×10^{-4}	1	266	537	117.2	21
1	0	0	1	-	-	-	-
3	0	0	1	-	-	-	-
	Linear demand function						
10^{-10}	2.94064×10^{7}	-1.49546×10^{15}	10^{-8}	263	858	126.5	21
10^{-6}	2.93488×10^{7}	-1.49190×10^{11}	10^{-4}	260	868	123.4	20
10^{-4}	2.41990×10^{7}	-1.18542×10^{9}	10^{-2}	119	604	47.3	19
10^{-1}	0	0	1	266	537	118.0	22
1	0	0	1	266	537	118.1	22
3	0	0	1	266	537	118.1	22
Barcelona							
	Constant-elasticity demand function						
10^{-10}	3.89399×10^{7}	-6.03680×10^{14}	10^{-7}	162	483	43.8	31
10^{-6}	3.89127×10^{7}	-6.03743×10^{10}	10^{-3}	156	481	43.5	30
10^{-4}	3.63590×10^{7}	-6.08433×10^{8}	10^{-1}	81	577	26.1	21
10^{-1}	5.61160×10^{-1}	-6.12803×10^{-1}	1	254	513	105.1	30
1	0	0	1	-	-	-	-
3	0	0	1	-	-	-	-
	Linear demand function						
10^{-10}	3.89399×10^{7}	-1.24366×10^{16}	10^{-9}	160	479	43.8	32
10^{-6}	3.89008×10^{7}	-1.24245×10^{12}	10^{-5}	160	476	43.6	33
10^{-4}	3.52103×10^{7}	-1.12974×10^{10}	10^{-3}	112	514	35.5	24
10^{-1}	0	0	1	254	513	103.7	29
1	0	0	1	254	513	102.2	28
3	0	0	1	254	513	103.6	29

Table 4: Results with the linear congestion function

References

[1] F. Babonneau, O. du Merle, and J.-P. Vial. Solving large scale linear multicommodity flow problems with an active set strategy and Proximal-ACCPM. *Operations Research*, 54(1):184–197, 2006.

[2] F. Babonneau and J.-P. Vial. ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. Technical report, HEC/Logilab, University of Geneva, 40 Bd du Pont d'Arve, CH-1211, 2005. To appear in *Mathematical Programming.*

[3] H. Bar-Gera. Origin-based algorithm for traffic assignment problem. *Transportation Science*, 36(4):398–417, 2002.

[4] J. Castro, L. Montero, and D. Rosas. Using ACCPM in a simplicial decomposition algorithm for the traffic assignment problem. Technical report, Statistics and Operations Research Department, Universitat Politecnica de Catalunya, 2002.

[5] S. Dafermos. Traffic equilibrium and variational inequalities. *Transportation Science*, 14(1):42–54, 1980.

[6] S.C. Dafermos and F.T. Sparrow. The traffic assignment problem for a general network. *J. Res. National Bureau of standards*, 73B:91–118, 1969.

[7] M. Daneva and P.O. Lindberg. The stiff is moving - conjugate direction Frank-Wolfe methods with applications to traffic assignment. Technical report, Linkoping University, Department of Mathematics, 2004.

[8] E.W. Dijkstra. A note on two problems in connexion with graphs. *Numer. Math.*, 1:269–271, 1959.

[9] S. Evans. Derivation and analysis of some models for combining trip distribution and assignment. *Transportation Research*, 10:37–57, 1976.

[10] M. Florian and S. Nguyen. A method for computing network equilibrium with elastic demands. *Transportation Science*, 8:321–332, 1974.

[11] H. Frank and P. Wolfe. An algorithm for quadratic programming. *Naval Research Logistics Quarterly*, 3:95–110, 1956.

[12] N. H. Gartner. Optimal traffic assignment with elastic demands: a review. part I. analysis framework. *Transportation Science*, 14(2):174–191, 1980.

[13] N. H. Gartner. Optimal traffic assignment with elastic demands: a review. part II. algorithmic approaches. *Transportation Science*, 14(2):192–208, 1980.

[14] J.L. Goffin, J. Gondzio, R. Sarkissian, and J.P. Vial. Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. *Mathematical Programming*, 76:131–154, 1996.

[15] J.-L. Goffin, A. Haurie, and J.-P. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38:284–302, 1992.

[16] D. W. Hearn and M. B. Yildirim. A toll pricing framework for traffic assignment problems with elastic demand. Technical report, Center for Applied Optimization Industrial and Systems Engineering Department, University of Florida, 2002 to appear in the edited volume: *Current trends in transportation and network analysis.*

[17] M. Josefsson and M. Patriksson. Sensitivity analysis of separable traffic equilibrium equilibria, with application to bilevel optimization in network design. *report, Department of Mathematics, Chalmers University of technology, Gotheburg*, 2003.

[18] G. Karakostas and S. G. Kolliopoulos. Edge pricing of multicommodity networks for selfish users with elastic demands. Technical report, Department of Computing & Software, McMaster University, 1280 Mian street West, Hamilton, Ontario, Canada, 2006.

[19] T. Larsson and M. Patriksson. An augmented Lagrangean dual algorithm for link capacity side constrainted traffic assignment problems. *Transportation Research*, 29B:433–455, 1995.

[20] T. Larsson and Di Yuan. An augmented Lagrangian algorithm for large scale multicommodity routing. *Computational Optimization and Applications*, 27(2):187–215, 2004.

[21] L.J. Leblanc and K. Farhangian. Efficient algorithms for solving elastic demand traffic assignment problems and mode split-assignment problems. *Transportation Science*, 15(4):306–317, 1982.

[22] A. Nagurney and D. Zhang. *Network equilibria and disequilibria, Equilibrium and Advanced Transportation Modeling*, pages 201–243. Kluwer Academic Publishers, Boston, MA, 1998.

[23] Y. Nesterov. Stable traffic equilibria: properties and applications. *Optimization and Engineering*, 3:29–50, 2000.

[24] A. Ouorou, H. P. L. Luna, and P. Mahey. Multicommodity network expansion under elastic demands. *Optimization and Engineering*, 2(3):277–292, 2001.

[25] M. Patriksson. *The traffic assignment problem: Models and Methods.* VSP, Utrecht, The Netherlands, 1994.

[26] J.G. Wardrop. Some theorical aspects of road traffic research. *Proceeding of the institute of civil engineers*, Part II:325–378, 1952.