Math. Program., Ser. B (2009) 120:179–210
DOI 10.1007/s10107-007-0151-3

FULL LENGTH PAPER

ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems

F. Babonneau · J.-P. Vial

Received: 6 September 2005 / Accepted: 30 October 2006 / Published online: 19 June 2007

Abstract This paper proposes an implementation of a constrained analytic center cutting plane method to solve nonlinear multicommodity flow problems. The new approach exploits the property that the objective of the Lagrangian dual problem has a smooth component with second order derivatives readily available in closed form. The cutting planes issued from the nonsmooth component and the epigraph set of the smooth component form a localization set that is endowed with a self-concordant augmented barrier. Our implementation uses an approximate analytic center associated with that barrier to query the oracle of the nonsmooth component. The paper also proposes an approximation scheme for the original objective. An active set strategy can be applied to the transformed problem: it reduces the dimension of the dual space and accelerates computations. The new approach solves huge instances with high accuracy. The method is compared to alternative approaches proposed in the literature.

Keywords Constrained ACCPM · Approximation scheme · Active set strategy

Mathematics Subject Classification (2000) Nondifferentiable optimization · Network flow problem

1 Introduction

The multicommodity network flow problem (MCF) consists of routing multiple commodities from a set of supply nodes to a set of demand nodes on a same underlying

F. Babonneau · J.-P. Vial (✉)
Logilab, HEC, Université de Genève, 40 Bd du Pont d'Arve,
1211 Geneva, Switzerland
e-mail: Jean-Philippe.Vial@hec.unige.ch

F. Babonneau
e-mail: fbabonneau@gmail.com

network. The cost associated with a routing is the sum of costs on the individual arcs. The cost on an individual arc is itself a function of the sum of the commodity flows on that arc. In the nonlinear multicommodity flow problem (NLMCF), the cost associated with an arc is a nonlinear (convex, increasing) function of the total flow on the arc, usually named *congestion* in transportation and *delay* in telecommunications. The main challenge of NLMCF is the size of problem instances. In this paper, we modify the Analytic Center Cutting Plane Method (ACCPM) in order to solve very large-scale instances.

NLMCF problems mainly arise in the areas of transportation and telecommunications. In the transportation sector, the concepts in use are "traffic assignment" and "Wardrop equilibrium". They are not equivalent, but both can be formulated as NLCMF, with arc costs that are convex polynomial functions of the flows. In the telecommunications sector, NLMCF models congestion on transmission networks. When several messages must be processed on a same link, the total processing time, hence the travel time of the messages, increases and tends to infinity when the flow approaches the arc capacity. This version of NLCMF is usually considered to be more difficult, due to the capacity constraint.

Two types of test problems can be found in the literature. In the first category, each commodity must be shipped from a single origin to a single destination. The potential number of commodities may be as large as the square of the number of nodes, a huge number on large networks, but finding the best route for a single commodity (independently of the other commodities) is a simple shortest path problem. In the second category, there are multiple supply nodes and demand nodes for each commodity. Finding the best route for a single commodity is then a transshipment problem, a more involved one, but the number of commodities is usually small to very small (with respect to the number of nodes). Papers in the literature deal with either one category, but not both. In this paper, we deal with problems in the first category.

The literature on NLMCF is abundant. We briefly review it. We first discuss methods that directly apply to the arc-flow formulation of the problem. The Frank and Wolfe method [9] works on a sequence of linearized versions of the problem. This approach is attractive because the direction finding subproblem is easy in the case of NLMCF problems with no capacity constraint. There, the direction finding subproblem turns out to be an unconstrained linear MCF that is separable in independent shortest path problems. The standard technique to cope with delay functions with a vertical asymptote consists in approximating the delay by a function with an unbounded domain. The main drawback of the Frank–Wolfe method is its slow convergence rate [2,26,31]. The Frank–Wolfe algorithm has been applied to NLMCF, e.g. [10,19]. The convergence of the method has been improved in [30]. More recently, Daneva and Lindberg [6] report dramatic acceleration using a conjugate gradient scheme.

Column generation is a standard tool to cope with large-scale optimization problems. In the case of NLMCF, this technique amounts to working on a sequence of restricted versions of the path-flow formulation of the problem. At each iteration, the method generates a shortest path for each commodity, with respect to arc lengths equal to the current marginal value of the delay or congestion function. The method then strives to allocate the flows on the generated paths in an optimal way. Bertsekas and Gafni [3] use a scaled projected Newton method to find an improved allocation. In the

comparative study carried in [25], the projected Newton method appeared to be one of the most efficient one. The method in [2] is an origin-based algorithm conceptually similar to [3]. Let us just mention that, as in the case of Frank–Wolfe, the method cannot handle arc capacity constraint directly. One must replace constraint violation by a fixed penalty function.

Most other methods can be classified as cutting plane methods (CPM). CPMs naturally arise in connection with a Lagrangian relaxation of the arc-flow formulation. It is easy to check that the relaxed master problem in a CPM is equivalent to the restricted primal in column generation. Lagrangian relaxation yields a Lagrangian dual problem of much smaller dimension. It is well known that the Lagrangian dual is nondifferentiable. Kelley's cutting plane method [14] is appealing because the master problem is linear, but it often converges very slowly. The bundle method has been used in the context of linear MCF [8] to remedy this drawback. The method could be extended to NLMCF. Good numerical results have been obtained with the analytic center cutting plane method [13]. In that paper, ACCPM was used in a disaggregate mode, that is with as many objective components as the number of commodities. An augmented Lagrangian technique has been used in [16] to solve non-linear traffic assignment problems with link capacity constraints. The subgradient method [27] is a possible alternative (see also [29]). It is easy to implement, but it is also known to converge very slowly. Recently, an important and promising enhancement has been proposed in [23]. To the best of our knowledge, this new method has not yet been applied to NLMCF. We conclude this brief review by mentioning the proximal decomposition method [20].

In this paper, we revisit ACCPM to improve its performance on transportation and telecommunications problems with nonlinear cost functions. The Lagrangian dual objective function of those problems has two main components: a piece-wise linear one (the same as in linear MCF) and one that is the negative Fenchel conjugate of the congestion function. The latter is smooth and can often be computed in closed form. In a traditional approach with ACCPM [13], the two components are approximated by cutting planes. The intersection of these half-spaces define a localization set whose analytic center becomes the point where to refine both approximations. In the present paper, we use in the definition of the localization set a direct representation of the epigraph of the smooth component as a fixed constraint. This approach is similar to [24] but our implementation does not use the embedding into a projective space.

The second contribution of this paper is an approximation scheme which replaces the congestion function near the origin by a linear function. This scheme is motivated by the fact that no Lagrangian dual variables need to be introduced in connection with arcs with a linear cost function. This results in a reduction of the dimension of the Lagrangian dual space and easier calculation of analytic center. This strategy has been implemented with success in [1] and can be described as an active set strategy aiming to find and eliminate unsaturated arcs. In the nonlinear case, the strategy is applied to arcs on which the optimal flow lies in a region where the cost function is well approximated by a linear function. The idea of active set has already been used in the context of linear multicommodity flow problems [1,8,21], but not within an approximation scheme for the nonlinear component in NLMCF.

The new method has been tested on standard problems that can be found in the open literature. We use four categories of problems. The first two categories, `planar` and `grid`, gather artificial problems that mimic telecommunication networks. Some of them are very large. The third category is made of four small to medium size telecommunication problems. One of them has been used in [25] to compare available specialized optimization codes. The last category includes six realistic traffic network problems; some of them are huge, with up to 13,000 nodes, 39,000 arcs and over 2,000,000 commodities.

The paper is organized as follows. In Sect. 2 we give formulation of the nonlinear multicommodity flow problem and in Sect. 3 we presents the associated Lagrangian relaxation. Section 4 provides a brief summary of the constrained ACCPM. Section 5 presents the most commonly used congestion functions and explicit their conjugate functions. Section 6 defines our approximation scheme. It discusses the active set and presents it as a partial Lagrangian relaxation. Section 7 details implementation issues and choices, while Sect. 8 is devoted to numerical results.

2 The nonlinear multicommodity flow problem

Let $\mathcal{G}(\mathcal{N}, \mathcal{A})$ be an oriented graph, where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ the set of arcs. The graph represents a network on which multiple commodities must be shipped from specific origins to specific destinations. Given a shipment schedule, the total flow on an arc induces a congestion.[1] The objective is to find a shipment schedule that minimizes the total congestion on the network. In many applications, in particular in transportation and telecommunications, the congestion function is nonlinear: the resulting problem is named the *nonlinear multicommodity flow problem*, in short NLMCF.

The arc-flow formulation of NLMCF is

$$\min\ g\left(\sum_{\tau\in\mathcal{T}} x^\tau\right) \equiv \sum_{a\in\mathcal{A}} g_a\left(\sum_{\tau\in\mathcal{T}} x_a^\tau\right) \tag{1a}$$

$$Nx^\tau = d_\tau \delta^\tau, \quad \forall \tau \in \mathcal{T}, \tag{1b}$$

$$x_a^\tau \geq 0, \quad \forall a \in \mathcal{A},\ \forall \tau \in \mathcal{T}. \tag{1c}$$

Here, N is the network incidence matrix; $\mathcal{T}$ is the set of commodities; d_τ is the demand for the commodity $\tau \in \mathcal{T}$; and δ^τ is vector of zeros except a "1" at the supply node and a "-1" at the demand node. The vector $x^\tau = (x_a^\tau)_{a\in\mathcal{A}}$ represents the flows of commodity τ on the arcs of the network. The number of arcs is $m = |\mathcal{A}|$; $|\mathcal{T}|$ is the number of commodities; and $|\mathcal{N}|$ is the number of nodes.

The congestion function $g : \mathbb{R}^m \to \mathbb{R}$ is assumed to be convex and twice continuously differentiable. The objective function is separable. Problem (1) has $n = |\mathcal{A}| \times |\mathcal{T}|$ variables and $|\mathcal{N}| \times |\mathcal{T}|$ constraints (plus possible upper bound constraints on variables). In general, $|\mathcal{T}|$ is large and $n \gg m > |\mathcal{N}|$. (We have in mind instances for which $m \geq 10^4$ and $n \geq 10^{10}$.)

[1] In the sequel, we shall not differentiate between "delay" and "congestion".

The literature essentially deals with two types of congestion function: the *Kleinrock* function, used in telecommunications [4,13,15,25] and the BPR (Bureau of Public Roads) function used in transportation [2,5,6,16]. The Kleinrock function is given by

$$g_a(y_a) = \frac{y_a}{c_a - y_a}, \quad \text{with } y_a \in [0, c_a), \tag{2}$$

where c_a is the capacity on the arc. The function has a vertical asymptote at $y_a = c_a$.

The BPR function is

$$g_a(y_a) = r_a y_a \left(1 + \frac{\alpha}{\beta + 1}(\frac{y_a}{c_a})^{\beta}\right), \quad \text{with } y_a \in \mathbb{R}^+. \tag{3}$$

In general, the parameter α is very small and $\beta > 1$ does not exceed 5. When the flow y_a is less than c_a, the second term under the parenthesis in (3) is negligible. Thus $g_a(y_a) \approx r_a y_a$: the parameter r_a is called *free-flow travel time* and 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 α and β are arc-dependent.

3 Lagrangian relaxation

For the sake of simpler notation, let us consider the more general problem

$$\min\{g(Mx) \mid x \in X\}. \tag{4}$$

We assume that $g : \mathbb{R}^m \to \mathbb{R}$ is convex, M is a $m \times n$ matrix, while $X \subset \mathbb{R}^n$ is convex. We easily identify problem (4) with (1). In nonlinear multicommodity flow problems on oriented graphs, X is defined as a set of network flow constraints (one per commodity). The matrix M collects the flows on the individual arcs; it is thus made of zeroes and ones. If the dimension of x is large, as it is the case in multicommodity flow problems, it becomes difficult to apply a direct method, even when the problem falls into the realm of structural programming (see [22]), e.g., g is self-concordant and X is endowed with a self-concordant barrier. An alternative to a direct method consists in applying a Lagrangian relaxation to a slightly transformed problem with an auxiliary variable y and an auxiliary constraint $y = Mx$. The new formulation

$$\begin{aligned} \min \quad & g(y) && \text{(5a)} \\ & Mx = y, && \text{(5b)} \\ & x \in X, && \text{(5c)} \end{aligned}$$

is equivalent to (4). Since the flows x on the oriented graph must be nonnegative, then $Mx \geq 0$. Therefore, the condition $y \geq 0$ is implied by (5b) and (5c). We need not introduce it explicitly in formulation (5). Relaxing (5b) yields a concave programming

problem in the dual variables $u \in \mathbb{R}^m$ associated with the constraints $y = Mx$. The Lagrangian dual problem is

$$\max_u L(u), \tag{6}$$

where L is defined by

$$L(u) = \min_{y,x}\{g(y) + \langle u, Mx - y\rangle \mid x \in X\}. \tag{7}$$

By duality, this Lagrangian dual problem has the same optimal value as (4). Since $m \ll n$, the transformed problem has much smaller dimension: it is potentially solvable by a suitable convex optimization method.

A quick inspection shows that (7) is separable in the variables y and x. Then, (7) can be written as

$$L(u) = f_1(u) + f_2(u), \tag{8}$$

where

$$f_1(u) = \min_x\{\langle M^T u, x\rangle \mid x \in X\},$$

and

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

In view of our assumption on X, $f_1(u)$ can be routinely computed for arbitrary values of u. Since f_1 is defined as the point-wise minimum of a collection of linear functions, f_1 is concave but usually nondifferentiable. Besides, if

$$x(u) = \operatorname{argmin}\{\langle M^T u, x\rangle \mid x \in X\},$$

then

$$f_1(u') \geq \langle Mx(u), u'\rangle = f_1(u) + \langle Mx(u), u' - u\rangle, \quad \forall u' \in \mathbb{R}^m. \tag{9}$$

This shows that $Mx(u)$ is an antisubgradient of $f_1(u)$ at u, that is $Mx(u) \in -\partial(-f_1(u))$.

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 similar to (9). 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.

Let us introduce conditions that are relevant for our multicommodity network flow problem of interest. They are of considerable help in solving (6).

Condition 1 *The linear programming problem*

$$\min\{\langle c, x\rangle \mid x \in X\},$$

can be solved at low computational cost.

In other words, $f_1(u)$ can be computed routinely, without excessive burden on the overall algorithm.

Condition 2 *The congestion function $g(y)$ is separable, i.e., $g(y) = \sum_{i=1}^{m} g_i(y_i)$. The functions g_i are nonnegative, convex, monotonically increasing and $\operatorname{dom} g_i \subset \mathbb{R}_+$. Moreover, the convex conjugate $(g_i)^*$ can be computed in closed form.*

Let us explore an immediate consequence of Condition 2. The first order optimality conditions for problem (5) are

$$0 \in \partial g(y) - u, \tag{10}$$

$$M^T u \in -\mathcal{N}_X(x), \tag{11}$$

where $\mathcal{N}_X(x)$ is the normal cone of X at x. The right derivative $g'_+(y)$ of g at $y = 0$ is well defined. Since g is convex on $\operatorname{dom} g = [0, c)$, (with possibly $c = \infty$), then $\partial g(y)$ is monotone. Thus for $y \geq 0$, condition (10) implies that the constraint

$$u \geq g'_+(0), \tag{12}$$

is always met at the optimum. It is nevertheless convenient to introduce this redundant constraint in the formulation of problem (6).

4 Constrained ACCPM

We aim to solve (6) with a version of ACCPM in which the smooth component f_2 of the objective function is handled as fixed, explicit constraint on the localization set. This constraint can be viewed as a cutting surface. The general setting is a problem as (6) with the constraint (12)

$$\max\{f(u) = f_1(u) + f_2(u) \mid u \geq u_l\}, \tag{13}$$

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 $\bar{u}$. This support takes the form of the cutting plane, called optimality cut

$$a^T(u - \bar{u}) + h(\bar{u}) \geq h(u), \quad \forall u \in \operatorname{dom} h, \tag{14}$$

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

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 $u = \bar{u}$.

We assume that f_1 is revealed by a first order oracle, while f_2 is accessed through a second-order oracle.

4.1 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 (14) 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, $k = 1, \ldots, K$, have been generated. The associated anti-subgradients $a^k \in -\partial(-f_1(u^k))$ 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

$$\mathcal{F}_{\underline{\theta}} = \{(u, z, \zeta) \mid -A^T u + ze \leq \gamma,\ \zeta \leq f_2(u),\ z + \zeta \geq \underline{\theta},\ u \geq u_l\}. \tag{15}$$

Clearly, the set contains all optimal pairs $(u^*, f(u^*))$. Thus, the search for a solution should be confined to the localization set.

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. For the sake of clarity, we first sketch the basic step, or *outer iteration*, of a generic cutting plane method.

```
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 f1.
3. Send the query point to the second order oracle to
   compute the objective function f2.
4. Update the lower and upper bounds and the
   localization set.
5. Test termination.
```

4.2 Proximal analytic centers

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

$$\min \quad F(u,z,\zeta) = \frac{\rho}{2}\|u-\underline{u}\|^2 - \sum_{i=0}^{K}\log s_i - \log\sigma - \sum_{i=1}^{m}\log(u_i - u_{li}) \quad (16a)$$

$$s_0 = z + \zeta - \underline{\theta} \geq 0, \quad (16b)$$

$$s_i = \gamma_i - z + (a^i)^T u \geq 0, \quad i = \{1,\ldots,K\}, \quad (16c)$$

$$\sigma = f_2(u) - \zeta \geq 0. \quad (16d)$$

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.

4.3 Newton method

If $f(x)$ is a self-concordant function with bounded level sets, then it is well known [22] that Newton method converges quadratically to the neighborhood

$$\left\{ x \mid \langle -[f''(x)]^{-1} f'(x),\, f'(x)\rangle \leq \eta < \frac{3-\sqrt{5}}{2} \right\}, \quad (17)$$

and that a damped Newton method converges to that neighborhood in a number of iterations that is polynomial. Thanks to a lemma in [12], it is easy to verify that the barrier function (16a) with f_2 equal to the Kleinrock or the BPR function is self-concordant.

In the rest of this subsection, we use the following notation. Given a vector $s > 0$, S is the diagonal matrix whose main diagonal is s. We also use $s^{-1} = S^{-1}e$ to denote the vector whose coordinates are the inverse of the coordinates of s. Similarly, $s^{-2} = S^{-2}e$. With this notation, the first order optimality conditions for Problem (16) are

$$\rho(u-\underline{u}) - As^{-1} - f_2'(u)\sigma^{-1} - (u-u_l)^{-1} = 0, \quad (18a)$$

$$e^T s^{-1} - s_0^{-1} = 0, \quad (18b)$$

$$\sigma^{-1} - s_0^{-1} = 0, \quad (18c)$$

$$s_0 - z - \zeta + \underline{\theta} = 0, \quad (18d)$$

$$s - \gamma + ze - A^T u = 0, \quad (18e)$$

$$\sigma - f_2(u) + \zeta = 0. \quad (18f)$$

The algorithm that computes the analytic center is a damped Newton method applied to (18). To write down the formulae, we introduce the residuals

$$\begin{aligned} r_u &= -(\rho(u-\underline{u}) - As^{-1} - f_2'(u)\sigma^{-1} - (u-u_l)^{-1}),\\ r_z &= -(e^T s^{-1} - s_0^{-1}),\\ r_\zeta &= -(\sigma^{-1} - s_0^{-1}),\\ r_{s_0} &= -(s_0 - z - \zeta + \underline{\theta}),\\ r_s &= -(s - \gamma + ze - A^T u),\\ r_\sigma &= -(\sigma - f_2(u) + \zeta). \end{aligned}$$

The Newton direction $(du, dz, d\zeta, ds_0, ds, d\sigma)$ associated to (18) is given by

$$P \begin{pmatrix} du \\ dz \\ d\zeta \\ ds_0 \\ ds \\ d\sigma \end{pmatrix} = \begin{pmatrix} r_u \\ r_z \\ r_\zeta \\ r_{s_0} \\ r_s \\ r_\sigma \end{pmatrix}, \tag{19}$$

where

$$P = \begin{pmatrix} \rho I - f_2(u)''\sigma^{-1} + (U-U_l)^{-2} & 0 & 0 & 0 & AS^{-2} & f_2(u)'\sigma^{-2} \\ 0 & 0 & 0 & s_0^{-2} & -e^T S^{-2} & 0 \\ 0 & 0 & 0 & s_0^{-2} & 0 & -\sigma^{-2} \\ 0 & -1 & -1 & 1 & 0 & 0 \\ -A^T & e & 0 & 0 & I & 0 \\ -f_2'(u) & 0 & 1 & 0 & 0 & 1 \end{pmatrix}.$$

Since (18f) is nonlinear, a full Newton step does not yield a feasible point with respect to (18f). Thus, we use the following empirical rule to compute the step length α_{step}. Let $0 < \gamma < 1$ be a fixed parameter and

$$\alpha_{\max} = \max\{\alpha \mid s + \alpha\, ds > 0,\ \sigma + \alpha\, d\sigma > 0,\ u + \alpha\, du > u_l\},$$

the step length is

$$\alpha_{step} = \min(1,\ \gamma\alpha_{\max}). \tag{20}$$

Since it is not essential in our solution method to compute an exact analytic center, we use the termination criteria $\eta = 0.99$ that is looser than (17).

To summarize, a basic step of the Newton iteration, or *inner iteration*, is

Inner iteration

1. Send the current point to the second order oracle to compute the objective function f_2 and its first and second derivatives.
2. Compute the Newton step $(du, dz, d\zeta, ds_0, ds, d\sigma)$ by (19).

`3. Compute a step length by (20) to update` $(u, z, \zeta, s_0, s, \sigma)$.
`4. Test termination.`

Remark 2 The computation in the inner iteration uses second order derivatives of f_2. Since the derivatives change at each iteration, the inner iteration must have access to the second order oracle. In pure cutting plane methods, the inner iteration does not interact with the oracle.

4.4 Upper and lower bounds

By duality, any feasible solution of (13) provides a lower bound for the original problem (1). Taking the values returned by the two oracles at the successive query points, we obtain the lower bound

$$\underline{\theta} = \max_{k \le K} \{f_1(u^k) + f_2(u^k)\}. \tag{21}$$

An upper bound $\bar{\theta}$ can be obtained from information collected in the computation of the analytic center. More precisely, assume that (u^c, z^c, ζ^c) is an approximate analytic center, in the sense that (s_0^c, s^c, σ^c), defined by (18d)–(18f), are strictly positive, and the equations (18a)–(18c) are approximately satisfied. Let

$$\lambda_c = (s_0^c)(s^c)^{-1} > 0. \tag{22}$$

If (18b) and (18c) are satisfied, then $e^T \lambda_c = 1$ and $(s_0^c)(\sigma^c)^{-1} = 1$. Otherwise, we scale λ_c and $(\sigma^c)^{-1}$ to meet the conditions. Using the cuts Mx^k, where $x^k = x(u^k)$, $k = 1, \ldots, K$ and $\bar{x}^K = \sum_{k=1}^K \lambda_c^k x^k$, we define the vector

$$\bar{y}^K = \sum_{k=1}^K \lambda_c^k M x^k = M \sum_{k=1}^K \lambda_c^k x^k = M \bar{x}^K. \tag{23}$$

The vector $\bar{x}^K$ is a convex combination of vectors in X: it also belongs to X. If $\bar{y}^K \in \operatorname{dom} g$ then, $g(\bar{y}^K)$ is a valid upper bound for the optimal value. If we extend the definition of g to have $g(y) = +\infty$ when $y \notin \operatorname{dom} g$, we have the upper bound

$$\bar{\theta} = \min_{k \le K} g(\bar{y}^k). \tag{24}$$

Let us now argue that we may expect that $\bar{y}^k$ becomes feasible. It is easy to relate

$$r = -\bar{y}^K - f_2'(u^c) - (s_0^c)(u^c - u_l)^{-1},$$

to the residual in (18a). If we are getting close to an optimal solution of the original problem, i.e., $\|u^c - u^*\|$ is small, we can reasonably hope that r is small with respect to $(s_0^c)(u^c - u_l)^{-1}$. Then, since $(s_0^c)(u^c - u_l)^{-1} \ge 0$ we have

$$0 \le \bar{y}^K \le -f_2'(u),$$

where $-f_2'(u^c) = (g^*)'(u^c) \in \operatorname{dom}(g)$. Thus, $\bar{y}^K$ is a feasible solution for (1).

5 Congestion functions and their conjugates

The implementation of ACCPM with a constraint requires an explicit calculation of the conjugate function g^* of the congestion g

$$g^*(u) = \max_y \{\langle u, y\rangle - g(y)\}.$$

In the sequel we shall name $-g^*$ the negative conjugate. The computation for the two congestion functions, Kleinrock and BPR, is straightforward. The objective $f_2(u)$ is separable into a the elementary functions g_a^*. We display in Table 1 the functional form of an individual component $-g_a^*$, its domain and its first and second derivatives. The functions and their conjugate are also plotted on Figs. 1 and 2.

In this paper, we shall also consider a more general class of congestion functions defined by

$$g_a(y_a) = \max_{y_a \in \mathrm{dom}\tilde{g}_a} \{t_a y_a, \tilde{g}_a(y_a)\}, \quad a \in \mathcal{A}, \tag{25}$$

where $t_a \geq 0$ and $\tilde{g}_a(y_a)$ is a convex nondecreasing function whose domain is a closed or half-closed interval of $\mathbb{R}_+$:

$$\mathrm{dom}\,\tilde{g}_a = [0, \bar{y}_a[\quad \text{or} \quad \mathrm{dom}\,\tilde{g}_a = [0, \bar{y}_a].$$

The upper limit may be finite (e.g., the support function associated with the constraint $y_a \leq \bar{y}_a$) or infinite. The meeting point between the linear and the nonlinear part is denoted y_a^c which is uniquely defined by $t_a y_a^c = \tilde{g}_a(y_a^c)$. In view of the convexity of g_a, we have $\tilde{g}_a'(y_a^c) \geq t_a$. We assume that $\tilde{g}_a$ is continuously differentiable on the interior of its domain. Note that $\mathrm{dom}\, g_a = \mathrm{dom}\,\tilde{g}_a$. We name the function (25) a *compound congestion* function.

Let us compute the negative conjugate of the compound congestion function. Let

$$y(u) = \arg\min_{y \in \mathrm{dom} g} \{g(y) - \langle u, y\rangle\},$$

Table 1 Conjugate functions

	Kleinrock delay function	BPR congestion function
Domain	$u_a \in [\frac{1}{c_a}, +\infty[$	$u_a \in [0, +\infty[$
Conjugate	$1 + u_a c_a - 2\sqrt{u_a c_a}$	$\frac{c_a (u_a - r_a)^{\frac{\beta+1}{\beta}}}{(\alpha r_a)^{\frac{1}{\beta}}} \frac{\beta}{\beta+1}$
Gradient	$c_a - \sqrt{\frac{c_a}{u_a}}$	$\frac{c_a}{(\alpha r_a)^{\frac{1}{\beta}}} (u_a - r_a)^{\frac{1}{\beta}}$
Hessian (diagonal)	$0.5\sqrt{c_a} u_a^{-1.5}$	$\frac{c_a}{\beta (\alpha r_a)^{\frac{1}{\beta}}} (u_a - r_a)^{\frac{1-\beta}{\beta}}$

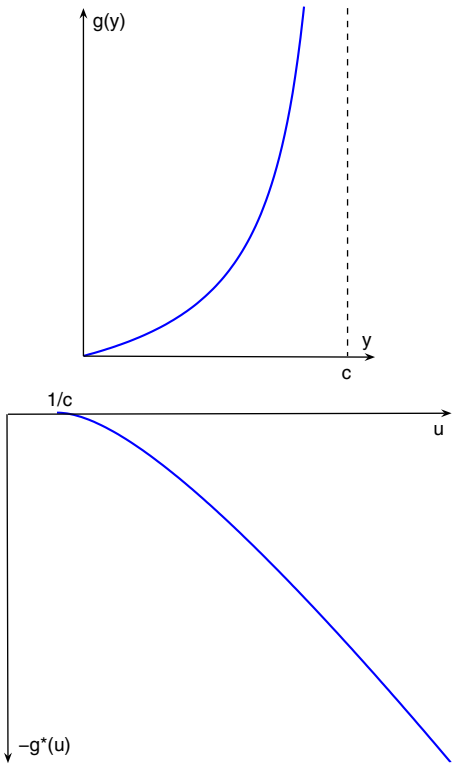

Fig. 1 Kleinrock delay function and its negative conjugate

whenever the minimum occurs in dom $\tilde{g}$. Simple calculation yields

$$-g_a^*(u_a) = \begin{cases} (t_a - u_a)y_a^c, & \text{with } y_a(u_a) = y_a^c, \quad \text{if } t_a \le u_a < \tilde{g}_a'(y_a^c), \\ -\tilde{g}_a^*(u_a), & \text{with } y_a(u_a) = [g_a']^{-1}(u_a), \quad \text{if } \tilde{g}_a'(y_a^c) \le u_a \le g_a'(\bar{y}_a). \end{cases}$$

It follows that dom $g^* \subset \{u \ge t\}$. In other words, we can add the constraint $u \ge t$ in the maximization of the Lagrangian dual function.

Finally, $g^*(u)$ is differentiable on the interior of dom $g^* \subset \{t < u < g'(\bar{y})\}$ and

$$(g^*)'(u) = y(u).$$

Figures 3 and 4 display the plot of compound functions and their negative conjugate in the case of Kleinrock and BPR functions, respectively.

6 Active set strategy on compound functions

Our interest for the class of compound congestion functions (25) has been triggered by the observation that multicommodity flow problems with linear congestion functions and no capacity constraint are easy to solve. Those problems are separable in the

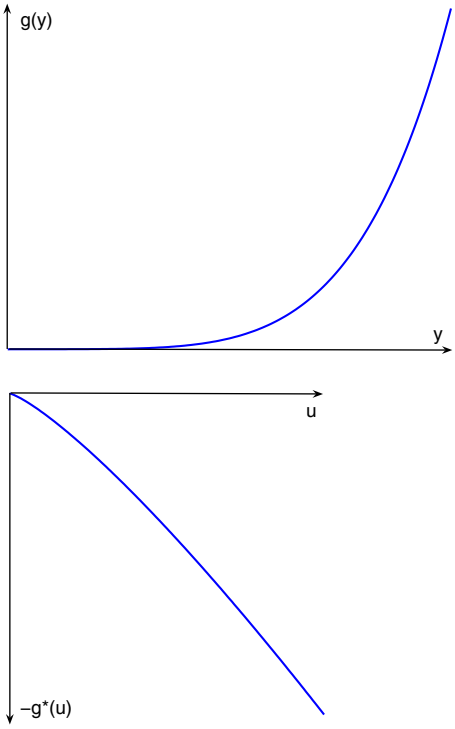

Fig. 2 BPR congestion function and its negative conjugate

commodities, and for each commodity the minimization problem boils down to a shortest path calculation. In the case of capacity constraints on the arcs, the above property can still be exploited on those arcs where the capacity constraint is inactive at the optimum. A heuristic that dynamically estimates the sets of active and inactive arcs has proved successful for LMCF [1]. This procedure can be extended to NLMCF problems with a compound congestion since, on those arcs where $y_a < y_a^c$, the function is linear. It can be further extended to standard NLMCF problems if one approximates the objective function by a compound one. Note that the approximation error is easily controlled by an appropriate choice of the meeting point y_a^c in (25). The approximation error tends to zero when $t_a \downarrow (\tilde{g}_a)'_+(0)$. In the previous section, we gave the formula for the negative conjugate of the compound congestion function. We can also use this expression to compute the maximal error on the dual side.

Consider the first order optimality conditions for the Lagrangian dual problem (6). In full generality, the condition stipulates that $u \geq u_l$ is optimal if there exists a nonnegative vector τ such that

$$\tau + \partial f_1(u) + \partial f_2(u) \ni 0, \tag{26}$$

$$\langle \tau, u - u_l \rangle = 0, \tag{27}$$

$$\tau \geq 0, \quad u \geq u_l, \tag{28}$$

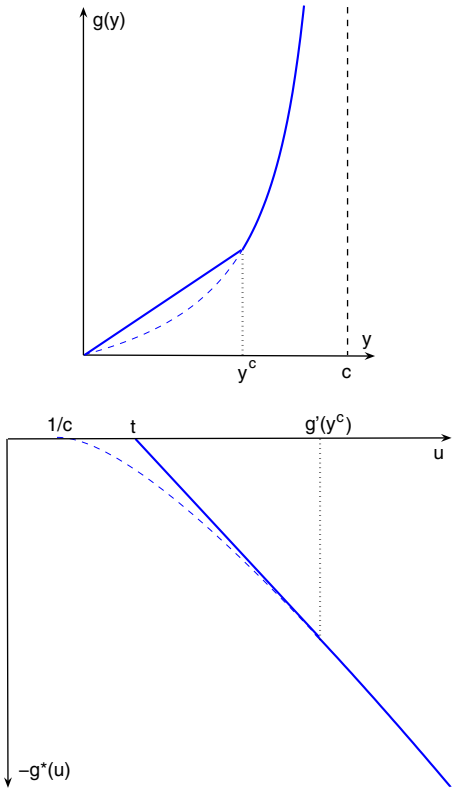

Fig. 3 Compound Kleinrock function and its conjugate

where $u_l = g'_+(0)$. An anti-subgradient of f_1 is of the form Mx with $x \in X$, while for f_2 one can take $-y(u)$. Hence, (26) and (27) imply

$$Mx \leq y(u),$$

and

$$(Mx)_a < y_a(u_a) \;\Rightarrow u_a = u_{la} = (g_a)'_+(0). \tag{29}$$

A similar analysis can be performed on the primal side.

Our goal is to use condition (29) to find a set $\mathcal{A}_1^* \subset \mathcal{A}$ with the property that for a optimal primal-dual pair (τ^*, u^*)

$$\tau_a^* > 0 \quad \text{and} \quad u_a^* = u_{la}, \;\; \forall a \in \mathcal{A}_1^*.$$

If the set $\mathcal{A}_1^*$ were known in advance, the variables u_a, $a \in \mathcal{A}_1^*$, could be fixed to their lower bound u_{la}. The original problem (6) would then boil down to a simpler problem in the variables u_a, $a \in \mathcal{A}_2^* = \mathcal{A} \setminus \mathcal{A}_1^*$. Note that $f_{2a}(u_{la}) = 0$; thus, the nonlinear term $f_{2a}(u_a)$, $a \in \mathcal{A}_1^*$, can be removed from this equivalent formulation. The above reasoning applies to any subset $\tilde{\mathcal{A}}_1 \subset \mathcal{A}_1^*$ and its complement $\tilde{\mathcal{A}}_2 \supset \mathcal{A}_2^*$.

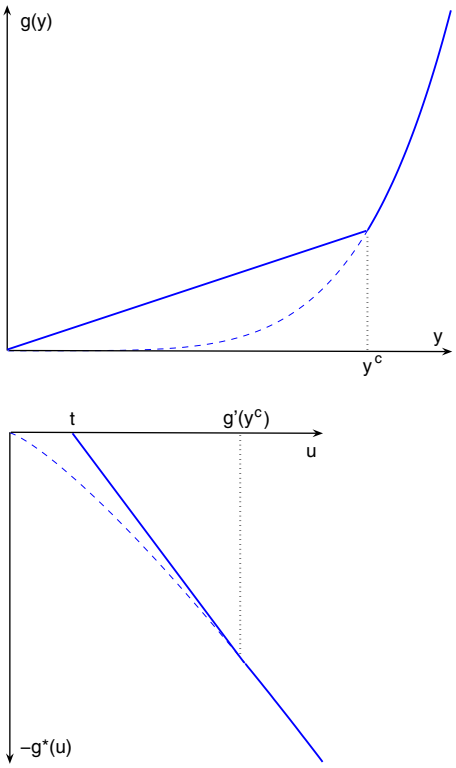

Fig. 4 Compound BPR function and its conjugate

In view of the above partition, we define (6) as the partial Lagrangian problem

$$\max\{L(u) \mid u_a = u_{la},\ a \in \mathcal{A}_1^*;\ u_a \geq u_{la},\ a \in \mathcal{A}_2^*\}.$$

In practice, the partition is not known in advance and the proposed space reduction technique cannot be straightforwardly implemented. An active set strategy aims to guess the partition. Let $\mathcal{A}_1$ and $\mathcal{A}_2$ be the current estimate of $\mathcal{A}_1^*$ and $\mathcal{A}_2^*$ with $\mathcal{A}_1 \cup \mathcal{A}_2 = \mathcal{A}$. This partition is used to work in a dual space of reduced dimension, a powerful gimmick if the cardinality of $\mathcal{A}_2^*$ is small. How one should build these approximation sets? The danger is to have an arc that moves from a set to its complement back and forth. We propose heuristic rules to avoid this bad behavior.

Suppose that our iterative procedure has generated Mx^k, $k = 1, \ldots, K$, with $x^k \in X$. From (23), we know that it is always possible to construct flows y^K that meet all the demands. We also recall that y^c is the meeting point between the linear and the nonlinear part. Assuming we are given a current partition of $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ into an active set and its complement, the rules that move elements between $\mathcal{A}_1$ and $\mathcal{A}_2$ are:

– An arc $a \in \mathcal{A}_1$ such that $y_a^K > y_a^c$ is moved into the active set $\mathcal{A}_2$.
– An arc $a \in \mathcal{A}_2$ such that $y_a^K \leq \gamma y_a^c$ is moved into the non active set $\mathcal{A}_1$.

Note that we introduce the parameter $\gamma < 1$ to increase the chances that an arc that is made inactive at some stage will not become active later in the process. In practice, we get $\gamma = 0.9$.

In the experiments, we approximate the Kleinrock delay function and the BPR congestion function by compound functions and use an active set strategy. The algorithm generates an ϵ-optimal primal-dual solution (y^*, u^*) for the compound function. The primal-dual pair can also be used to measure the relative optimality gap with the original functions $\tilde{g}$ and $\tilde{g}^*$. This gap depends on the quality of the approximation by the compound function and thus may be larger than ϵ.

7 Implementation issues

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

7.1 First order oracle

The first order oracle consists of $|\mathcal{T}|$ shortest path computations, using Dijkstra's algorithm [7]. 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. This defines a subset of nodes $\mathcal{S} \subset \mathcal{N}$. We apply $|\mathcal{S}|$ times Dijkstra's algorithm, once for each $s \in \mathcal{S}$. 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 would most likely improve the performance of the overall algorithm, but the focus of the paper is on the cutting plane method and not on shortest path computation.

7.2 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; the weight on the logarithmic barrier on the floor cut; and a heuristic to eliminate almost inactive cutting planes.

7.2.1 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 initial value for the proximal coefficient ρ is 1. The rule to update this parameter is the following. When the method do not improve the lower bound $\underline{\theta}$ during few iterations, it may happen that the weight of the generated cuts is to large pushing the query point to far from the proximal reference point. To fix this behavior, we increase the impact of the proximal term multiplying ρ by 10 to ensure the new query point

to remain closest from the best recorded value. It thus makes it easier for ACCPM to find a best dual solution, i.e., a best lower bound.

7.2.2 Weight on floor cut

The localization set is bounded below by the special constraint $z + \zeta \geq \underline{\theta}$ in (15). 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 (16). 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.

7.2.3 Column elimination strategy

It is well known that column generation techniques are adversely affected by the total number of generated columns. This is particularly true with ACCPM, since the Newton iterations in the computation of the analytic center have a complexity that is roughly proportional to the square of the number of generated columns. It is thus natural to try to reduce the total number of columns by eliminating irrelevant elements. Our criterion to eliminate columns is based on the contribution of a column to the primal flow solution. Let λ be defined by Eq. (22). (We assume without loss of generality that $e^T\lambda = 1$.) Then $A\lambda$ is the total flow on the network. If λ_i is much smaller than the average of λ, then column i contributes little to the solution (dually, the distance s_i between the analytic center and the cut is high,) and is a good candidate for elimination. To make the elimination test robust, we compute the median of the λ's and eliminate any column whose coefficient λ_i is less than $1/\kappa$ times the median. In practice, we choose $\kappa = 4$. We also perform the test once every $\tau = 20$ iterations. (For the largest problem, we took $\tau = 100$.)

7.3 Termination criterion

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

$$(\bar{\theta} - \underline{\theta})/\max(\underline{\theta}, 1) \leq \epsilon, \tag{30}$$

where $\underline{\theta}$ is the best lower bound computed with (21) and $\bar{\theta}$ is the best upper bound computed with (24). In our experiments we use $\epsilon = 10^{-5}$.

7.4 Approximation scheme

The goal of the approximation scheme is to replace the original nonlinear function $\tilde{g}_a$ by the compound function g_a defined by (25). The error function induced by the

linear approximation

$$e_a(y_a) = t_a y_a - \tilde{g}_a(y_a), \quad y_a \in [0, y_a^c].$$

Let $\hat{y}_a \in [0, y_a^c]$ be the point that maximizes the error e_a. Let $\mathcal{A}_1^*$ be the set of inactive arcs at the optimum. The error due to the approximation is bounded by $\sum_{a \in \mathcal{A}_1^*} e_a(\hat{y}_a)$. We want that this error to be lower than $\epsilon \tilde{g}^*$, where ϵ is the relative optimality gap and $\tilde{g}^*$ is the optimal objective value. Furthermore we impose that $e_a(\hat{y}_a) = \mu$ be the same for all $a \in \mathcal{A}_1^*$. We estimate μ by

$$\mu = \frac{\hat{\epsilon}\tilde{g}^*}{|\mathcal{A}_1^*|}, \quad \text{with } \hat{\epsilon} < \epsilon. \tag{31}$$

Then we compute t_a such that $e_a(\hat{y}_a) = \mu$. Unfortunately $\tilde{g}^*$ and $|\mathcal{A}_1^*|$ must be estimated. In the experiments, we take the parameter $\hat{\epsilon} = 10^{-6}$ and $|\mathcal{A}_1^*| = n/2$. The value of $\tilde{g}^*$ is chosen empirically depending on the class of problems.

8 Numerical experiments

The main goal of our empirical study is to test the efficiency (i) of using a nonlinear cutting surface, (ii) of column elimination, (iii) of an active set strategy and (iv) of a joint implementation of a column elimination scheme and an active set strategy. We also use published results to benchmark the new algorithm.

8.1 Test problems

We used four sets of test problems. The first set, the `planar` problems, contains ten instances that have been generated by Di Yuan to simulate telecommunication problems. Nodes are randomly chosen as points in the plane, and arcs link neighbor nodes in such a way that the resulting graph is planar. Commodities are pairs of origin and destination nodes, chosen at random. Demands and capacities are uniformly distributed in given intervals. The second set, the `grid` problems, contains 15 networks that have a grid structure such that each node has four incoming and four outgoing arcs. Note that the number of paths between two nodes in a grid network is usually large. Commodities, and demands are generated in a way similar to that of `planar` networks. These two sets of problems are used to solve the linear multicommodity flow problem in [1,17]. The data include arc capacities and linear costs and can be downloaded from http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html. We use directly these arc capacities in the definition of the Kleinrock function. To solve (1) with BPR function, we use the capacity as *practical capacity* and the linear cost as *free-flow travel time*. As suggested in [28], we use the parameter values $\alpha = 0.15$ and $\beta = 4$.

The third collection of problems is composed of telecommunication problems of various sizes. The small problems `ndo22` and `ndo148` are two practical problems

solved in [11,13]. Problem 904 is based on a real telecommunication network and was used in the survey paper [25]. This problem set is adapted to solve (1) with Kleinrock function. To solve (1) with BPR function, we use the capacity as *practical capacity* and also use it as *free-flow travel time*. We choose the parameter values $\alpha = 0.15$ and $\beta = 4$.

The last collection of problems is composed of six realistic transportation problems used in [1,2,6,16]. Some of them are huge, with up to 13,000 nodes, 39,000 arcs and over 2,000,000 commodities. The data are adapted for the BPR function. They include *free-flow travel time*, *practical capacity* and the tuning parameters α and β. These problems, can be downloaded from http://www.bgu.ac.il/~bargera/tntp/. To solve (1) with Kleinrock function we use *practical capacity* as capacity and to turn these problems feasible with respect to the capacity, which is handled by the objective function, the demands are reduced as in [1,16].

Table 2 displays data on the four sets of problems. For each problem instance, we give the number of nodes $|\mathcal{N}|$, the number of arcs $|\mathcal{A}|$, the number of commodities $|\mathcal{K}|$, the optimal solution values to (1) $z^*_{\text{Kleinrock}}$ for the Kleinrock function and z^*_{BPR} for the BPR function, with a relative optimality gap less than 10^{-5}.

8.2 Numerical results

We carry the experiments with the two congestion functions: the Kleinrock delay function and the BPR congestion function. For each solution strategy, we attempt to solve all problem instances contained in Table 2 with a 10^{-5} relative optimality gap. To benchmark the results with our best solution strategy, we use, for telecommunications problems (Kleinrock function), the results with the Projection Method reported in [25] and, for transportation problems (BPR function) several implementations of Frank–Wolfe algorithm reported in [6].

For all results using ACCPM, the tables give 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. When the active set strategy is activated, the working space of ACCPM is reduced to the active arcs only. Thus, we give the percentage of arcs in the active set, $\%|\mathcal{A}_2|$, at the end of the solution process. We display also the error, denoted *Error*, resulting from the approximation with respect to the optimal solution of the original problem. Finally, when the elimination column is activated we display the number of remaining cuts *Nb cuts* at the end of the process.

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.

8.2.1 Impact of using a nonlinear cutting surface

In this subsection, we experiment the impact of the second order information in the solution method solving all the instances. We compare ACCPM using a second order

Table 2 Test problems

Problem ID	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{K}\rvert$	$z^*_{\text{Kleinrock}}$	z^*_{BPR}
`planar` problems					
`planar30`	30	150	92	40.5668	4.44549×10^7
`planar50`	50	250	267	109.478	1.21236×10^8
`planar80`	80	440	543	232.321	1.81906×10^8
`planar100`	100	532	1,085	226.299	2.29114×10^8
`planar300`	300	1,680	3,584	329.120	6.90748×10^8
`planar500`	500	2,842	3,525	196.394	4.83309×10^9
`planar800`	800	4,388	12,756	354.008	1.16952×10^9
`planar1000`	1,000	5,200	20,026	1, 250.92	3.41859×10^9
`planar2500`	2,500	12,990	81,430	3, 289.05	1.23827×10^{10}
`grid` problems					
`grid1`	25	80	50	66.4002	8.33599×10^5
`grid2`	25	80	100	194.512	1.72689×10^6
`grid3`	100	360	50	84.5618	1.53241×10^6
`grid4`	100	360	100	171.331	3.05543×10^6
`grid5`	225	840	100	236.699	5.07921×10^6
`grid6`	225	840	200	652.877	1.05075×10^7
`grid7`	400	1,520	400	776.566	2.60669×10^7
`grid8`	625	2,400	500	1, 542.15	4.21240×10^7
`grid9`	625	2,400	1,000	2, 199.83	8.36394×10^7
`grid10`	625	2,400	2,000	2, 212.89	1.66084×10^8
`grid11`	625	2,400	3,000	1, 502.75	3.32475×10^8
`grid12`	900	3,480	6,000	1, 478.93	5.81488×10^8
`grid13`	900	3,480	12,000	1, 760.53	1.16933×10^9
`grid14`	1,225	4,760	16,000	1, 414.39	1.81297×10^9
`grid15`	1,225	4,760	32,000	1, 544.15	3.61568×10^9
Telecommunication-like problems					
`ndo22`	14	22	23	11.5631	1.87110×10^3
`ndo148`	58	148	122	151.926	1.40233×10^5
`904`	106	904	11,130	33.4931	1.29197×10^7
Transportation problems					
`Sioux-Falls`	24	76	528	600.679	4.23133×10^6
`Winnipeg`	1,067	2,975	4345	1, 527.41	8.25673×10^5
`Barcelona`	1,020	2,522	7922	845.872	1.23277×10^6
`Chicago-sketch`	933	2,950	93,513	615.883	1.67484×10^7
`Chicago-region`	12,982	39,018	2,297,945	3, 290.55	2.58457×10^7
`Philadelphia`	13,389	40,003	1,151,166	2, 558.01	1.27810×10^8

oracle and ACCPM in which the smooth function is handled implicitly by the first order oracle as in the traditional approach.

The results are reported in Table 3 for the Kleinrock function and in Table 4 for the BPR function. In the two cases, we observe that the new approach outperforms the classical ACCPM. The larger problems are not solved by the classical ACCPM, partly because too many cuts in the localization set jammed the memory space.

8.2.2 Impact of column elimination

In this subsection, ACCPM solves the sets of problems using the column elimination. We report the results in Table 5 for the Kleinrock function and in Table 6 for the BPR function. The last column *CPU Ratio* displays the improvement ratio of the CPU time of ACCPM without column elimination (see Tables 3 and 4), and with column elimination. We observe that column elimination speed-up the CPU time on all problems, with an average value 1.5. Since the number of outer iterations is about the same, the speed-up is due to a reduction of the computation time in ACCPM. It is apparent in comparing the proportion of time spent in the oracle.

8.2.3 Impact of the active set strategy

In this subsection, we experiment the active strategy to solve (1) with the Kleinrock and BPR functions. This strategy turns out to be efficient only with the BPR function, but not with the Kleinrock function. The very steep slope of Kleinrock close its to asymptote leads to a larger spread of the flows on the arcs. All arcs turn out to be moderately congested and the compound function does not provide a satisfactory approximation.

Table 7 gives the computational results using the active set strategy on the approximate BPR function. The last column, shows the improvement ratio of CPU time of ACCPM without active set strategy (see Table 4), and with the active set strategy. The value of g^* in (31) is empirical. We get $g^* = 10^8$ for telecommunication instances (`planar`, `grid` and telecommunications-like problems) and $g^* = 10^7$ for traffic networks.

Table 7 shows that active set strategy speed-up the CPU time on all problems (excepted the smaller ones) until 8.4. This speed-up is partly due to the large number of inactive arcs in the optimal solution. The number of dual variables handled by ACCPM is usually lower than 60%. A second explanation is the reduction of the total number of outer iteration around 10%. Removing the inactive arcs from the Lagrangian relaxation seems to make ACCPM easier the converge. The important point is that the quality of the optimal solution is not affected by the approximation, i.e., the computed optimal solution for the approximate problem is also a optimal solution with 10^{-5} optimality gap for the original problem. For three instances, the approximation does not ensure a 10^{-5} optimality gap but it is also traduced by a larger decrease of number of outer iterations, of size of the active set, and obviously of CPU time. This observation shows the difficulties to guaranty a given optimality gap in a static approximation scheme.

Table 3 Impact of the cutting surface (Kleinrock delay function)

Problem ID	ACCPM with cutting surface				ACCPM without cutting surface			
	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
planar30	93	197	1.1	22	832	1,664	59.2	17
planar50	134	279	2.8	20	1,234	2,468	253.5	14
planar80	182	383	8.1	16	1,965	3,930	1, 381.3	11
planar100	187	392	10.2	17	2,342	4,684	2, 593.5	10
planar300	175	367	29.5	24	–	–	–	–
planar500	127	324	32.2	37	–	–	–	–
planar800	182	429	110.5	40	–	–	–	–
planar1000	381	869	568.1	26	–	–	–	–
planar2500	543	1,224	3, 471.7	45	–	–	–	–
grid1	52	118	0.4	24	462	924	11.5	21
grid2	93	212	1.0	25	456	912	11.4	21
grid3	138	341	4.1	15	1,713	3,426	798.4	12
grid4	167	344	5.7	17	1,613	3,226	710.2	12
grid5	204	474	18.5	17	3,409	6,818	11, 005.7	8
grid6	333	686	55.9	14	3,326	6,652	10, 457.4	8
grid7	410	811	155.4	15	–	–	–	–
grid8	845	1,783	1, 416.8	10	–	–	–	–
grid9	582	1,269	576.9	15	–	–	–	–
grid10	432	964	300.6	20	–	–	–	–
grid11	261	581	106.4	29	–	–	–	–
grid12	201	409	106.7	41	–	–	–	–
grid13	222	454	128.7	39	–	–	–	–
grid14	204	414	173.2	48	–	–	–	–
grid15	203	414	172.8	48	–	–	–	–
ndo22	12	86	0.2	7	173	346	1.5	27
ndo148	70	361	1.3	7	737	1,474	45.7	17
904	135	294	10.4	27	–	–	–	–
Sioux-falls	140	345	1.7	24	533	1,410	13.0	15
Winnipeg	338	988	215.0	14	–	–	–	–
Barcelona	253	678	101.1	15	–	–	–	–
Chicago-sketch	145	370	48.6	41	–	–	–	–
Chicago-region	190	500	8, 621.9	94	–	–	–	–
Philadelphia	279	822	13, 094.4	89	–	–	–	–

8.2.4 Impact of active set strategy with column elimination

In this set of experiments, we combine both column elimination and the active set strategy. Of course, since active set strategy is not efficient with Kleinrock function, we

Table 4 Impact of the cutting surface (BPR congestion function)

Problem ID	ACCPM with cutting surface				ACCPM without cutting surface			
	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
planar30	56	256	1.2	25	202	502	5.7	15
planar50	96	422	3.4	15	328	775	21.8	11
planar80	159	665	13.0	12	651	1,400	165.0	7
planar100	101	423	6.5	16	629	1,341	172.3	8
planar300	98	358	18.3	21	1,101	2,398	1, 980.2	6
planar500	42	164	10.0	37	986	2,261	2, 548.9	7
planar800	88	299	51.1	43	2,155	4,513	32, 792.7	5
planar1000	192	552	209.6	37	–	–	–	–
planar2500	364	1,744	3, 099.2	38	–	–	–	–
grid1	24	108	0.4	26	148	329	1.8	30
grid2	49	202	0.8	25	238	557	3.5	27
grid3	31	121	0.7	23	259	586	13.4	16
grid4	57	216	1.7	22	386	859	35.3	10
grid5	60	199	3.5	23	557	1,230	174.2	7
grid6	125	385	11.5	16	918	1,931	660.0	6
grid7	102	307	15.1	22	1,111	2,405	1, 872.9	5
grid8	158	422	49.5	23	1,982	4,119	13, 010.7	3
grid9	211	597	97.3	22	2,379	4,923	22, 202.1	3
grid10	207	586	94.8	24	2,404	4,965	22, 925.9	3
grid11	138	413	47.9	31	1,966	4,082	12, 873.5	4
grid12	107	323	52.7	46	2,002	4,369	20, 391.3	4
grid13	117	340	59.1	44	2,300	4,785	28, 038.6	4
grid14	84	274	61.9	56	1,995	4,179	26, 729.3	5
grid15	93	293	70.2	55	1,588	4,011	17, 449.6	4
ndo22	4	35	0.1	0	75	287	0.8	23
ndo148	6	45	0.2	0	171	390	3.6	24
904	93	316	8.4	19	802	1,729	470.2	6
Sioux-falls	80	411	2.0	19	366	1,057	10.1	20
Winnipeg	81	298	16.3	36	1,307	2,783	3, 352.8	8
Barcelona	56	245	10.4	29	925	2,040	1, 493.7	8
Chicago-sketch	72	265	20.4	48	1,828	4,075	11, 891.0	5
Chicago-region	332	1,502	10, 606.5	64	–	–	–	–
Philadelphia	287	1,250	7, 469.9	63	–	–	–	–

solve only (1) with the BPR congestion function. The settings of the active set strategy and the column elimination are those used in the previous subsections. The results are displayed on Table 8. In the last column of the table, we give the improvement ratio

Table 5 Impact of column elimination (Kleinrock delay function)

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	ratio
planar30	54	114	293	1.1	26	1.0
planar50	67	164	385	2.2	27	1.3
planar80	126	239	544	6.5	24	1.2
planar100	95	208	481	6.0	24	1.7
planar300	104	205	484	22.2	32	1.3
planar500	75	129	319	24.1	47	1.3
planar800	99	179	447	77.1	54	1.4
planar1000	207	369	836	303.6	43	1.9
planar2500	339	567	1,292	2, 398.5	64	1.5
grid1	45	59	194	0.5	27	0.8
grid2	55	108	242	0.8	46	1.0
grid3	71	121	307	2.3	22	1.8
grid4	71	188	394	3.1	31	1.8
grid5	137	203	472	12.0	21	1.5
grid6	113	389	828	24.3	26	2.3
grid7	244	471	1,024	90.9	22	1.7
grid8	311	876	1,845	384.0	21	3.7
grid9	344	646	1,397	305.5	24	1.9
grid10	274	474	1,039	199.8	29	1.5
grid11	228	270	599	96.6	32	1.1
grid12	189	212	431	107.2	43	1.0
grid13	207	234	478	125.8	41	1.0
grid14	183	212	430	166.5	51	1.0
grid15	181	205	418	161.5	52	1.1
ndo22	12	12	86	0.2	7	1.0
ndo148	47	73	176	0.8	25	1.6
904	85	138	300	7.6	27	1.4
Sioux-falls	73	144	353	1.6	25	1.1
Winnipeg	202	384	1,155	135.3	19	1.6
Barcelona	182	261	732	71.2	19	1.4
Chicago-sketch	77	130	340	30.1	53	1.6
Chicago-region	92	194	597	8, 672.1	95	1.0
Philadelphia	109	290	826	13, 021.8	93	1.0

of the CPU time of ACCPM without using the two options (see Table 4), and with using them. As expected, column elimination reduces the computational time. As in Sect. 8.2.2, it decreases the time spent in computing the analytic centers.

Table 6 Impact of column elimination (BPR congestion function)

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	ratio
planar30	26	57	276	1.2	21	1.0
planar50	34	95	418	2.9	23	1.2
planar80	39	167	611	6.7	21	1.9
planar100	25	99	383	4.3	23	1.5
planar300	42	98	345	12.6	33	1.5
planar500	24	40	156	8.6	44	1.2
planar800	36	76	250	34.2	56	1.5
planar1000	72	177	519	135.1	51	1.5
planar2500	120	346	1,350	1, 657.9	66	1.9
grid1	15	24	105	0.4	33	1.0
grid2	27	52	194	0.8	33	1.0
grid3	20	30	114	0.7	27	1.0
grid4	20	55	194	1.5	28	1.1
grid5	30	61	194	2.4	9	1.5
grid6	39	139	380	6.8	24	1.7
grid7	34	100	283	9.2	32	1.6
grid8	51	166	441	30.6	35	1.6
grid9	57	217	578	46.4	40	2.1
grid10	52	206	560	45.6	44	2.1
grid11	39	144	398	30.0	48	1.6
grid12	35	100	287	34.5	60	1.5
grid13	37	122	346	44.2	60	1.3
grid14	28	88	270	53.1	67	1.2
grid15	33	99	293	59.9	68	1.2
ndo22	4	4	35	0.1	0	1.0
ndo148	6	6	45	0.2	0	1.0
904	37	100	311	6.2	27	1.4
Sioux-falls	28	83	346	1.4	30	1.4
Winnipeg	36	76	259	10.6	47	1.5
Barcelona	20	55	231	7.9	38	1.3
Chicago-sketch	36	84	272	18.3	57	1.1
Chicago-region	124	342	1,327	8, 224.8	85	1.2
Philadelphia	95	254	1,031	4, 962.5	83	1.5

8.2.5 Comparisons with other methods

In this subsection, we compare ACCPM with a Projection Method on telecommunications problems using the Kleinrock delay function. We also compare ACCPM with

Table 7 Impact of the active set strategy (BPR congestion function)

Problem ID	Error	$\%\|A_2\|$	Outer	Inner	CPU	$\%Or$	ratio
planar30	$<10^{-5}$	53	58	299	1.2	36	1.0
planar50	$<10^{-5}$	63	99	473	2.9	33	1.2
planar80	$<10^{-5}$	62	154	803	9.4	24	1.4
planar100	$<10^{-5}$	60	97	536	4.9	29	1.3
planar300	$<10^{-5}$	44	82	399	9.2	38	2.0
planar500	$<10^{-5}$	26	38	216	5.8	62	1.7
planar800	$<10^{-5}$	26	77	383	29.6	68	1.7
planar1000	$<10^{-5}$	40	168	775	132.2	54	1.6
planar2500	$<10^{-5}$	43	323	1,996	2, 063.3	51	1.5
grid1	$<10^{-5}$	86	25	118	0.4	32	1.0
grid2	$<10^{-5}$	99	49	210	0.8	31	1.0
grid3	$<10^{-5}$	40	25	122	0.6	33	1.1
grid4	$<10^{-5}$	50	52	222	1.7	34	1.0
grid5	$<10^{-5}$	47	50	224	2.3	37	1.5
grid6	$<10^{-5}$	63	107	400	8.6	31	1.3
grid7	$<10^{-5}$	52	76	312	8.8	39	1.7
grid8	$<10^{-5}$	54	113	437	26.1	38	1.9
grid9	$<10^{-5}$	64	178	612	60.8	34	1.6
grid10	$<10^{-5}$	66	195	661	74.5	34	1.3
grid11	$<10^{-5}$	61	139	475	41.9	43	1.1
grid12	$<10^{-5}$	51	87	330	34.6	59	1.5
grid13	$<10^{-5}$	57	113	448	53.4	51	1.1
grid14	$<10^{-5}$	44	78	323	47.2	69	1.3
grid15	$<10^{-5}$	49	88	346	56.9	66	1.2
ndo22	$<10^{-5}$	50	5	46	0.1	−10	1.0
ndo148	$<10^{-5}$	75	6	49	0.2	17	1.0
904	$<10^{-5}$	32	93	358	5.1	42	1.6
Sioux-falls	$<10^{-5}$	100	69	336	1.7	30	1.2
Winnipeg	1.3×10^{-5}	33	48	246	6.8	49	2.4
Barcelona	$<10^{-5}$	24	37	241	3.8	51	2.7
Chicago-sketch	$<10^{-5}$	52	65	262	16.9	56	1.2
Chicago-region	5×10^{-4}	34	55	698	1, 261.3	87	8.4
Philadelphia	5×10^{-5}	54	97	1,234	2, 157.9	74	3.5

several implementations of Frank–Wolfe algorithm on transportation problems with the BPR congestion function.

Table 8 Active set strategy and column elimination (BPR congestion function)

Problem ID	Error	$\%\|A_2\|$	Nb cuts	Outer	Inner	CPU	%Or	ratio
planar30	$<10^{-5}$	53	26	58	315	1.2	30	1.0
planar50	$<10^{-5}$	63	26	92	475	2.9	31	1.2
planar80	$<10^{-5}$	63	42	196	896	7.6	33	1.7
planar100	$<10^{-5}$	60	36	98	506	4.2	32	1.5
planar300	$<10^{-5}$	44	35	96	434	9.4	50	1.9
planar500	$<10^{-5}$	26	16	34	196	5.1	70	2.0
planar800	$<10^{-5}$	27	31	78	363	27.1	77	1.9
planar1000	$<10^{-5}$	20	78	177	815	113.6	64	1.9
planar2500	$<10^{-5}$	43	118	354	1,898	1, 540.0	73	2.0
grid1	$<10^{-5}$	86	16	25	117	0.4	34	1.0
grid2	$<10^{-5}$	97	21	56	229	0.8	31	1.0
grid3	$<10^{-5}$	40	18	26	125	0.7	33	1.0
grid4	$<10^{-5}$	50	27	51	197	1.4	40	1.2
grid5	$<10^{-5}$	47	22	45	207	1.9	39	1.8
grid6	$<10^{-5}$	63	47	115	384	6.4	41	1.8
grid7	$<10^{-5}$	52	37	73	290	6.7	47	2.3
grid8	$<10^{-5}$	54	45	116	413	19.1	50	2.6
grid9	$<10^{-5}$	64	45	177	680	39.0	48	2.5
grid10	$<10^{-5}$	67	42	188	605	40.9	55	2.3
grid11	$<10^{-5}$	61	44	139	455	28.4	59	1.7
grid12	$<10^{-5}$	51	32	89	321	29.4	70	1.8
grid13	$<10^{-5}$	57	35	112	418	38.9	64	1.5
grid14	$<10^{-5}$	44	33	74	324	40.8	75	1.5
grid15	$<10^{-5}$	49	35	84	328	47.4	75	1.5
ndo22	$<10^{-5}$	50	5	5	46	0.1	−12	1.0
ndo148	$<10^{-5}$	75	6	6	49	0.1	23	2.0
904	$<10^{-5}$	32	37	114	358	3.5	55	2.4
Sioux-falls	$<10^{-5}$	100	31	93	413	1.5	40	1.3
Winnipeg	1.4×10^{-5}	33	21	47	251	5.7	51	2.9
Barcelona	$<10^{-5}$	24	18	35	213	3.1	54	3.3
Chicago-sketch	$<10^{-5}$	52	29	68	269	15.1	64	1.3
Chicago-region	5×10^{-4}	34	22	55	681	1, 229.5	89	8.6
Philadelphia	5×10^{-5}	54	41	99	1,185	2, 004.0	81	3.7

Table 9 Test problems

Problem ID	Load factor	$z^*_{\text{Kleinrock}}$
904	1	33.4931
904(1.5)	1.5	52.2678
904(2)	2	72.6437
904(2.5)	2.5	94.8839
904(3)	3	119.305

Table 10 ACCPM and PM

Problem ID	ACCPM					Previous ACCPM[a]			PM[a]		
	Nb cuts	Outer	Inner	CPU	%Or	Outer	CPU	ratio	Outer	CPU	ratio
904	85	138	300	7.6	27	14	3,233	30.4	579	380	3.6
904(1.5)	105	150	314	9.8	24	14	3,441	25.0	663	434	3.2
904(2)	109	172	357	11.8	24	14	3,186	19.3	688	471	2.8
904(2.5)	107	189	398	13.9	24	13	3,276	16.8	741	558	2.9
904(3)	101	192	415	12.6	24	13	3,544	20.1	691	501	2.8

[a]Tests performed in [25] on an IBM RISC/System 6000 machine

ACCPM versus the Projection Method

In this experiment, we compare the results of our solution method ACCPM using column elimination with the results of the Projection Method (PM) reported in [25]. As in [25], we solve problem 904 with a varying load factor to generate different demands. Table 9 gives the load factors we use and the corresponding optimal value with a 10^{-5} relative optimality gap.

In [25], the authors compare a previous version of ACCPM implemented in [13] with the Flow Deviation Method [18], the Projection Method [4], and the Proximal Decomposition Method [20]. In this comparative study, the Projection Method (PM) appears to be the most efficient method to solve the 904 instances. We use the figures reported in [25] for PM and ACCPM.[2]

The computational tests in [25] are performed on an IBM RISC/System 6000. We report the original computing times in Table 10. In order to compare these results with those we obtain with the new version of ACCPM on a Pentium IV, we have performed benchmark computations according to BYTEmark.[3] We found a ratio 14. We use this value to compare the speeds of the algorithms in the two columns entitled *ratio* in Table 10. These ratios are just indicative.

[2] The ACCPM version [13] works on a disaggregated form of the objective function. It exploits the sparsity in the master problem to cope with the very large number of generated cuts. In the disaggregated approach, the oracle generates as many cuts as the number of commodities at each outer iteration. In the case of problem 904, this means 11,130 cuts.

[3] BYTE Magazine's BYTEmark benchmark program (release 2) available at http://www.byte.com/bmark/bmark.htm.

Table 11 ACCPM vs. Frank–Wolfe

Problem ID	ACCPM	BFW	CFW	FW
Sioux-falls	44	124	357	1,869
Winnipeg	39	163	243	838
Barcelona	31	41	34	51
Chicago-sketch	27	21	17	24
Chicago-region	115	43	53	126

Table 10 shows that the new ACCPM using column elimination outperforms the previous version of ACCPM and improves the computational time of the Projection Method with a ratio 3.

ACCPM versus Frank–Wolfe algorithm

In this experiment, we compare ACCPM with the results obtained in [6] with different versions of Frank–Wolfe algorithm: a classical Frank–Wolfe method (FW), a conjugate direction Frank–Wolfe method (CFW) and a bi-conjugate Frank–Wolfe method (BFW). These methods outperform the Frank–Wolfe method implemented in [2]. We solve the same set of transportation problems as in [6], with BPR function with a 10^{-4} relative gap.

Since we have not at our disposal the machine used in [6], we cannot compare the computational times. To get an idea of ACCPM performances, we focus on the number of iterations to solve the problems to compare ACCPM to the different versions of FW. In this experiment, we do not use the active set strategy to have perfect control on the precision of the optimal solution. The results displayed on Table 11 show that ACCPM is competitive with the implementations of Frank–Wolfe algorithm in term of number of iterations, except for the last instance. ACCPM is more efficient on the smaller instances.

9 Conclusion

In this paper, we proposed two important modifications of the analytic center cutting plane method to solve nonlinear multicommodity flow problems: a cutting surface to handle the smooth component of the Lagrangian dual objective and an approximation scheme for the nonsmooth component of that objective. The approximation scheme is coupled with an active set strategy that leads to an expression of the Lagrangian dual in a space of smaller dimension. The new approach considerably improves the performance of the former implementation of ACCPM. It compares favorably with the known most efficient methods.

The computation in a Lagrangian relaxation approach breaks into two main components: computation of the query point (an approximate analytic center of the localization set, in our case) and the solving of the oracle subproblem (shortest paths computation, in our case). The overall computation depends on the computational efficiency of these two operations and on the number of times they are performed, i.e., on the number of calls to the oracle. In this paper, we focused on two items: the

number of outer iterations and the computation of the query point. In contrast, our implementation of the shortest path algorithm is rather straightforward and could be improved, in particular if one wishes to exploit special network structures.

The present study suggests that possible further improvements could be achieved using the approximation/active set approach with a different linearization scheme for the cost function. Conceptually, this linearization could be performed around points that are dynamically chosen to lead more efficient approximations. This will be the object of further researches.

Acknowledgments The work was partially supported by the Fonds National Suisse de la Recherche Scientifique, grant # 12-57093.99.

References

1. Babonneau, F., du Merle, O., Vial, J.-P.: Solving large scale linear multicommodity flow problems with an active set strategy and Proximal-ACCPM. Oper. Res. **54**(1), 184–197 (2006)
2. Bar-Gera, H.: Origin-based algorithm for traffic assignment problem. Transp. Sci. **36**(4), 398–417 (2002)
3. Bertsekas, D.P., Gafni, E.M.: Projected newton methods and optimization of multicommodity flows. IEEE Trans. Autom. Control **28**, 1090–1096 (1983)
4. Bertsekas, D.P., Gallager, R.G.: Data Networks. Prentice-Hall, Englewood Cliffs (1987)
5. Castro, J., Montero, L., Rosas, D.: Using ACCPM in a simplicial decomposition algorithm for the traffic assignment problem. Technical report, Statistics and Operations Research Department, Universitat Politecnica de Catalunya (2002)
6. Daneva, M., Lindberg, P.O.: The stiff is moving—conjugate direction Franck–Wolfe methods with applications to traffic assignment. Technical report, Linkoping University, Department of Mathematics (2004)
7. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numer. Math. **1**, 269–271 (1959)
8. Frangioni, A., Gallo, G.: A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. Inf. J. Comput. **11**, 370–393 (1999)
9. Frank, H., Wolfe, P.: An algorithm for quadratic programming. Naval Res. Logist. Q. **3**, 95–110 (1956)
10. Fratta, L., Gerla, M., Kleinrock, L.: The flow deviation method: an approach to store-and-forward communication network design. Networks **3**, 97–133 (1973)
11. Gafni, E.M., Bertsekas, D.P.: Two-metric projection methods for constrained optimization. SIAM J. Control Optim. **22**(6), 936–964 (1984)
12. Glineur, F.: Improving complexity of structured convex optimization problems using self concordant barriers. Eur. J. Oper. Res. **143**, 291–310 (2002)
13. Goffin, J.-L., Gondzio, J., Sarkissian, R., Vial, J.-P.: Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. Math. Program. **76**, 131–154 (1996)
14. Kelley, J.E.: The cutting plane method for solving convex programs. J. SIAM **8**, 703–712 (1960)
15. Kleinrock, L.: Nets, Stochastic Message Flow and Delay, Communications. Dover, New York (1972)
16. Larsson, T., Patriksson, M.: An augmented lagrangean dual algorithm for link capacity side constrained traffic assignment problems. Transp. Res. **29**, 433–455 (1995)
17. Larsson, T., Di, Yuan: An augmented lagrangian algorithm for large scale multicommodity routing. Comput. Optim. Appl. **27**(2), 187–215 (2004)
18. Leblanc, L.: Mathematical programming algorithms for large scale network equilibrium and network design problems. Ph.D. thesis, IE/MD Dept, Northwestern University, Evanston IL (1973)
19. Leblanc, L., Helgason, R.V., Boyce, D.E.: Improved efficiency of the Frank–Wolfe algorithm for convex network programs. Transportation Science (1985)
20. Mahey, P., Ouorou, A., Leblanc, L., Chifflet, J.: A new proximal decomposition algorithm for routing in telecommunication networks. Networks **31**(4), 227–238 (1998)
21. McBride, R.D.: Progress made in solving the multicommodity flow problem. SIAM J. Optim. **8**(4), 947–955 (1998)

22. Nesterov, Y.: Introductory Lectures on Convex Optimization: A Basic Course. Kluwer, Dordrecht (2004)
23. Nesterov, Y.: Smooth minimization of non-smooth functions. Math. Program. **103**(1), 127–152 (2005)
24. Nesterov, Y., Vial, J.-P.: Homogeneous analytic center cutting plane methods for convex problems and variational inequalities. SIAM J. Optim. **9**(3), 707–728 (1999)
25. Ouorou, A., Mahey, P., Vial, J.-P.: A survey of algorithms for convex multicommodity flow problems. Manage. Sci. **46**, 126–147 (2000)
26. Patriksson, M.: The Traffic Assignment Problem—Models and Methods. VSP, Utrecht (1994)
27. Poljak, B.T.: Subgradient methods: a survey of Soviet research. In: Lemaréchal, C., Mifflin, R. (eds.) Nonsmooth Optimization, pp. 5–30. Pergamon Press, Oxford (1977)
28. Sheffi, Y.: Urban Transportation Networks: Equilibrium Analysis with Mathematical Programming Models. Prentice-Hall, New Jersey (1985)
29. Shor, N.: Nondifferentiable optimization and polynomial problems. Kluwer, Boston (1998)
30. Weintraub, A., Ortiz, C., Gonzales, J.: Accelerating convergence of the Franck-Wolfe algorithm. Transp. Res. **19**, 113–122 (1985)
31. Wolfe, P.: Convergence theory in nonlinear programming. In: Abadie, J. (ed.) Integer and Nonlinear Programming, pp. 1–36 (1970)