Solving large scale linear multicommodity flow problems with an active set strategy and proximal-accpm

Babonneau, Frédéric; Du Merle, Olivier; Vial, Jean-Philippe

FACULTE DES SCIENCES ECONOMIQUES ET SOCIALES

HAUTES ETUDES COMMERCIALES

SOLVING LARGE SCALE LINEAR MULTICOMMODITY FLOW PROBLEMS WITH AN ACTIVE SET STRATEGY AND PROXIMAL-ACCPM

F. BABONNEAU
O. DU MERLE
J.-P. VIAL

HEC GENÈVE

UNIVERSITÉ DE GENÈVE

Solving large scale linear multicommodity flow problems with an active set strategy and Proximal-ACCPM

F. Babonneau[†] O. du Merle[*] J.-P. Vial[†]

June, 2004

Abstract

In this paper, we propose to solve the linear multicommodity flow problem using a partial Lagrangian relaxation. The relaxation is restricted to the set of arcs that are likely to be saturated at the optimum. This set is itself approximated by an active set strategy. The partial Lagrangian dual is solved with Proximal-ACCPM, a variant of the analytic center cutting plane method. The new approach makes it possible to solve huge problems when few arcs are saturated at the optimum, as it appears to be the case in many practical problems.

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

1 Introduction

The instances of the linear multicommodity flow problem, in short LMCF, that are encountered in real applications are often of such large a size that it is impossible to solve them as standard linear programming problems. To get around the challenge of dimension, one often resorts to Lagrangian relaxation. One has then to solve the Lagrangian dual problem, a non-differentiable concave programming problem of much smaller size. We propose an efficient way to handle the dual problem and get a primal feasible solution with guaranteed precision.

The literature on the linear multicommodity flow problem proposes many different solution methods. Direct approaches consist in solving the large linear programming problem with a linear programming code exploiting the special block-network structure of the constraint matrix. The solver can be either simplex-based, e.g. [13], or use interior point methodology, e.g. [1]. The other popular approach is based on price-directive decomposition, i.e., a Lagrangian relaxation approach, or equivalently, a column generation scheme. The standard way to solve the Lagrangian dual problem is Kelley's method [10], or its dual, the Dantzig-Wolfe decomposition scheme [3]. Farvolden et al. [6] apply Dantzig-Wolfe decomposition to solve an arc-chain formulation of the multicommodity flow problem. In [2], P. Chardaire and A. Lisser compare Dantzig-Wolfe decomposition with direct approaches and also with ACCPM [9] on oriented multicommodity flow problems. They conclude that using a fast simplex-based linear programming code to solve the master program in Dantzig-Wolfe decomposition is the fastest alternative on small

[*]Air France, Operation Research Department, 1 avenue du Maréchal Devaux, 91550 Paray-Vielle-Poste.
[†]Logilab, HEC, Université de Genève, 40 Bd du Pont d'Arve, CH-1211 Geneva, Switzerland.

and medium size problems. The bundle method [12] can also be used to solve the Lagrangian dual. A. Fragioni and G. Gallo [7] implement a bundle type approach. In a recent contribution, Larsson and Yuang [11] apply an augmented Lagrangian algorithm that combines Lagrangian relaxation and nonlinear penalty technique. They introduce nonlinear penalty in order to improve the dual convergence and obtain convergence to a primal solution. They are able to find solutions with reasonable precision on very large problem instances. They also compare their method to Dantzig-Wolfe decomposition and to bundle method.

In this paper, we apply the analytic center cutting plane method [9] to solve the Lagrangian dual problem. Following [5], we add a proximal term to the barrier function of the so-called localization set. We further speed up the solution method by using an active set strategy. This is motivated by our observation that on practical problems, the number of congested arcs in an optimal solution is but a small fraction of the total number of arcs in the graph. In [2], P. Chardaire and A. Lisser claim that in real transmission networks the number of saturated arcs is usually no more than 10 percent of the total number of arcs. T. Larsson and Di Yuan in [11] make the same observation on their test problems. McBride and Mamer report the same in [14]. In other words, for a large majority of arcs, the total flow in the optimal solution is strictly less than the installed capacity. Consequently, the Lagrangian dual variables associated with these arcs must be null at the optimum. If this (large) set of null optimal dual variables were known in advance, one could perform a partial Lagrangian relaxation restricted to the saturated arcs. This would considerably reduce the dimension of the Lagrangian dual and make it much easier to solve. In practice, the set of saturated arcs at the optimum is not known, but can be dynamically estimated as follows. An arc is added to the active set as soon as the flow associated with the current primal solution exceeds the capacity of this arc. The strategy to remove an arc from the active set is more involved. The arcs to be discarded are selected among those whose capacity usage by the current solution is below some threshold, but a dual-based safeguard is implemented to decrease the chances that a deleted arc be later reinserted in the active set. The use of an active set strategy in solving the general multicommodity flow problem is not new. It has been implemented within the framework of bundle method to solve the Lagrangian dual [7] and in a primal partitioning method [15]. Both papers report significant speed-ups.

The new method is applied to solve a collection of linear multicommodity problems that can be found in the open literature. As in [11], we focus on problems with many commodities, each one having one origin and one destination node. The subproblems are simple shortest path problems, possibly very numerous. 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. 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. We are able to find an exactly feasible solution with a relative optimality gap less that 10^{-5} for all problems including the larger ones.

The paper is organized as follows. In Section 2 we give formulations of the linear multicommodity flow problem. Section 3 presents Lagrangian relaxations. In Section 4 we define our so-called active set that amounts a partial Lagrangian relaxation. Section 5 deals with a brief summary of ACCPM and its variant with a proximal term. Section 6 details our algorithm and provides information on its implementation. Section 7 is devoted to the numerical experiments.

2 The linear multicommodity flow problem

Given a network represented by the directed graph $\mathcal{G}(\mathcal{N}, \mathcal{A})$, with node set $\mathcal{N}$ and arc set $\mathcal{A}$, the arc-flow formulation of the linear multicommodity flow problem is the following linear programming problem

$$\begin{aligned}
\min \quad & \sum_{a\in\mathcal{A}} t_a \sum_{k\in\mathcal{K}} x_a^k && && \text{(1a)}\\
& \sum_{k\in\mathcal{K}} x_a^k \le c_a, && \forall a \in \mathcal{A}, && \text{(1b)}\\
& Nx^k = d_k\delta^k, && \forall k \in \mathcal{K}, && \text{(1c)}\\
& x_a^k \ge 0, && \forall a \in \mathcal{A},\ \forall k \in \mathcal{K}. && \text{(1d)}
\end{aligned}$$

Here, N is the network matrix; t_a is the unit shipping cost on arc a; d_k is the demand for commodity k; and δ^k is vector with only two non-zeros components: 1 at the supply node and -1 at the demand node. The variable x^k is the flow of commodity k on the arcs of the network. The parameter c_a represents the capacity on arc $a \in \mathcal{A}$ to be shared among all commodity flows x_a^k. Problem (1) has $|\mathcal{A}| \times |\mathcal{K}|$ variables and $|\mathcal{A}| + |\mathcal{N}| \times |\mathcal{K}|$ constraints (plus the bound constraints on the variables). Let us mention that problem (1) can be formulated in a more compact way. Consider the set of commodities that share the same origin node. If we replace the constraints (1c) associated with those commodities with their sum, we obtain a new formulation with less constraints and variables. It can be easily checked that this new formulation is equivalent to (1). This transformation may be useful for direct methods, particularly on problems with many more commodities than nodes, because it drastically reduces the problem dimension. If the solution method is a Lagrangian relaxation, as it is the case in this paper, the transformation is irrelevant: the complexity of the problem remains unchanged.

The linear multicommodity flow problem can be also formulated using path-flows instead of arc-flows. Though we shall not really use this formulation, it may prove useful to remind it for the sake of illustration. Let us denote π a path on the graph from some origin to some destination. A path is conveniently represented by a Boolean vector on the set of arcs, with a "1" for arc a if and only if the path goes through a. For each commodity k, we denote $\{\pi_j\}_{j\in J_k}$ the set of paths from the origin of demand d_k to its destination. Finally, the flow on path π_j is denoted ξ_j and the cost of shipping one unit of flow along that path is γ_j. The extensive path-flow formulation of the problem is

$$\begin{aligned}
\min \quad & \sum_{k\in\mathcal{K}} \sum_{j\in J_k} \xi_j\gamma_j && && \text{(2a)}\\
& \sum_{k\in\mathcal{K}} \sum_{j\in J_k} \xi_j\pi_j \le c, && && \text{(2b)}\\
& \sum_{j\in J_k} \xi_j = d_k,\ \forall k \in \mathcal{K}, && && \text{(2c)}\\
& \xi \ge 0.
\end{aligned}$$

The path-flow formulation is compact but the cardinality of the set of paths is exponential with the dimension of the graph. Therefore, it cannot be worked out explicitly, but on very small problems. It is nevertheless useful, since the solution method to be discussed later can be interpreted as a technique of generating a few interesting paths and use them to define a restricted version of problem (2).

3 Lagrangian and partial Lagrangian relaxations

In the standard Lagrangian relaxation of (1), one relaxes the coupling capacity constraint $\sum_{k\in\mathcal{K}} x_a^k \le c_a$ and assigns to it the dual variable $u \ge 0$. The Lagrangian dual problem is

$$\max_{u\ge 0} \mathcal{L}(u)$$

where

$$\mathcal{L}(u) = \min_{x^k \geq 0,\ k \in \mathcal{K}} \left\{ L(x,u) \mid Nx^k = \delta^k,\ \forall k \in \mathcal{K} \right\}, \tag{3}$$

and $L(x,u)$ is the Lagrangian function

$$L(x,u) = \sum_{a \in \mathcal{A}} t_a \sum_{k \in \mathcal{K}} x_a^k + \sum_{a \in \mathcal{A}} u_a (\sum_{k \in \mathcal{K}} x_a^k - c_a).$$

Since the Lagrangian dual is the minimum of linear forms in u, it is concave. Moreover, it is possible to exhibit an element of the anti-subgradient of $\mathcal{L}$ at $\bar{u}$ if we know an optimal solution $\bar{x}$ of (3) at $\bar{u}$. Indeed, for any u, we have from the definition of $\mathcal{L}$

$$\mathcal{L}(u) \leq \sum_{a \in \mathcal{A}} t_a \sum_{k \in \mathcal{K}} \bar{x}_a^k + \sum_{a \in \mathcal{A}} u_a (\sum_{k \in \mathcal{K}} \bar{x}_a^k - c_a). \tag{4}$$

Inequality (4) clearly shows that $-c + \sum_{k \in \mathcal{K}} \bar{x}^k$ is an anti-subgradient. Inequality (4) is sometimes referred to as an *optimality cut* for $\mathcal{L}$. Note that the minimization problem in (3) is separable into $|\mathcal{K}|$ shortest path problems. We also recall that the optimal solutions u^* of the Lagrangian dual problem are also optimal solutions of the dual of the linear programming problem (1).

The dimension of the decision variable in $\mathcal{L}(u)$ is $|\mathcal{A}|$. We now investigate the possibility of reducing the problem size. From the strict complementarity theorem, we know that there exists at least a pair (x^*, u^*) of strictly complementary primal and dual solutions of (1). Actually we know more: there exists a unique partition $\mathcal{A} = \mathcal{A}_1^* \cup \mathcal{A}_2^*$ of the set of arcs such that for all strictly complementary pair

$$\begin{cases} u_a^* > 0 \quad \text{and} \quad \sum\limits_{k \in \mathcal{K}} (x_a^k)^* = c_a, & \forall a \in \mathcal{A}_1^*, \\ u_a^* = 0 \quad \text{and} \quad \sum\limits_{k \in \mathcal{K}} (x_a^k)^* < c_a, & \forall a \in \mathcal{A}_2^*. \end{cases}$$

This partition is called the *optimal partition*. If this partition were known in advance, it would possible to drop all constraints

$$\sum_{k \in \mathcal{K}} (x_a^k)^* \leq c_a, \quad \forall a \in \mathcal{A}_2^*,$$

in (1). This information would also be very useful in solving a Lagrangian relaxation of (1).

If $|\mathcal{A}_1^*|$ is much smaller than $|\mathcal{A}|$, the associate Lagrangian dual also has much smaller dimension. Since in practical problems it is generally observed that the number of inactive constraints is large, the strict complementarity theorem suggests that the dimension of the Lagrangian dual function can be dramatically reduced, and thus the problem made easier. In view of the above partition, we define the partial Lagrangian as

$$\begin{aligned} L_{\mathcal{A}_1^*}(x, u_{\mathcal{A}_1^*}) &= \sum_{a \in \mathcal{A}} t_a \sum_{k \in \mathcal{K}} x_a^k + \sum_{a \in \mathcal{A}_1^*} u_a (\sum_{k \in \mathcal{K}} x_a^k - c_a) \\ &= -\sum_{a \in \mathcal{A}_1^*} u_a c_a + \sum_{a \in \mathcal{A}_1^*} (t_a + u_a) \sum_{k \in \mathcal{K}} x_a^k + \sum_{a \in \mathcal{A}_2^*} t_a \sum_{k \in \mathcal{K}} x_a^k. \end{aligned}$$

The Lagrangian dual problem is

$$\max_{u_{\mathcal{A}_1^*} \geq 0} \{ \mathcal{L}_{\mathcal{A}_1^*}(u_{\mathcal{A}_1^*}) = -\sum_{a \in \mathcal{A}_1^*} u_a c_a + \mathcal{M}_{\mathcal{A}_1^*}(u_{\mathcal{A}_1^*}) \}, \tag{5}$$

with

$$\mathcal{M}_{\mathcal{A}_1^*}(u_{\mathcal{A}_1^*}) = \sum_{k\in\mathcal{K}} \min\left\{ \sum_{a\in\mathcal{A}_1^*} (t_a + u_a)x_a^k + \sum_{a\in\mathcal{A}_2^*} t_a x_a^k \mid Nx^k = \delta^k,\, x_a^k \geq 0, \quad \forall a \in \mathcal{A} \right\}.$$

Note that one can obtain an anti-subgradient of $\mathcal{M}_{\mathcal{A}_1^*}$ in the exact same manner as described for the full Lagrangian dual function (see (4)).

4 Active set strategy

Obviously, the optimal partition $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ is not known in advance. We shall now discuss ways of dynamically estimate it. The scheme is based on the fact that it is always possible to construct flows that meet all the demands. If the resulting total flow does not meet the capacity constraint on some arc in $\mathcal{A}_2$, then this arc is potentially binding at the optimum and should be moved to $\mathcal{A}_1$. A more complex scheme shifts arcs from $\mathcal{A}_1$ to $\mathcal{A}_2$. Prior to describing with full details the updating schemes, let us see how one can obtain the above-mentioned vector of total flows.

The Lagrangian relaxation scheme substitutes to problem (1) the dual problem (3) that is solved by a cutting plane method based on (4). In this inequality, $\bar{\Pi} = \sum_{k\in\mathcal{K}} \bar{x}^k$ is the total flow resulting from shipping the individual commodities along appropriate paths. Let us also denote $\bar{\Gamma} = \sum_{k\in\mathcal{K}} \sum_{a\in\mathcal{A}} t_a \bar{x}_a^k$ the cost associated with that flow. Suppose that our iterative procedure has generated Π_t, $t = 1, \ldots T$, such vectors with associated costs Γ_t. Any convex combination of such vectors, defines flows on the arcs

$$y = \sum_{t=1}^{T} \mu_t \Pi_t, \;\; \text{with} \;\; \sum_{i=1}^{t} \mu_t = 1,\, \mu \geq 0,$$

that can be associated with individual commodity flows that satisfy the demands. The issue is to check whether this total flow is compatible with the capacity constraints. We partition $\mathcal{A}$ with respect to this y to get $\mathcal{A}_1 = \{a \mid y_a \geq c_a\}$ and $\mathcal{A}_2 = \{a \mid y_a < c_a\}$. The set $\mathcal{A}_1$ could be used to estimate $\mathcal{A}_1^*$. In the sequel, we shall name it the *active set*.

The discussion raises the issue on how to find the above-mentioned convex combination vector. In the Lagrangian relaxation, the problem can be seen as the one of finding appropriate prices where to query the oracle. The constraints set consists in a set of inequalities generated by the oracle. The dual view of the problem consists in finding a convex combination of path-flow columns. This defines the cuts, or, in a column generation framework, a restricted version of the original path-flow formulation (2). It is easy to see that μ can be identified with a primal variable for the master program. It appears to be a by-product in Kelley's method when solving the master program. Proximal-ACCPM and others methods also generate this information. We thus can use it to form a convex combination of the cuts generated by the oracle.

We conclude this section by presenting a heuristic rule to update the active set in the course of the maximization of the Lagrangian dual function. Assume we are given a set of paths as described above and a current partition of $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ into an active set and its complement. Assume also we are given a dual variable u, with $u_{\mathcal{A}_1} \geq 0$ and $u_{\mathcal{A}_2} = 0$, and a set of non-negative variable μ_t summing to one. These variables form a primal dual pair of solutions in the restricted path-flow problem

$$\min \quad \sum_{t=1}^{T} \mu_t \Gamma_t \tag{6a}$$

$$(\sum_{t=1}^{T} \mu_j \Pi_t)_a \leq c_a,\, \forall a \in \mathcal{A}_1, \tag{6b}$$

$$\sum_{t=1}^{T} \mu_t = 1, \tag{6c}$$
$$\mu \geq 0.$$

The μ variable is feasible for the original path-flow problem if (6b) also holds for all $a \in \mathcal{A}_2$. If not, any arc in $\mathcal{A}_2$ such that $(\sum_{t=1}^{T} \mu_t \Pi_t)_a > c_a$ should be moved into the active set.

The rule to remove an arc from the active set is heuristic. Assume that the pair (μ, u) is reasonably closed to an optimal solution of (6). There are two obvious necessary conditions for an arc to be moved to the inactive set. First, the current flow on the arc should be sufficiently far away from the available capacity. Second, the Lagrangian dual variable u_a should be closed enough to zero. We have thus two threshold values $\eta_1 > 0$ and $\eta_2 > 0$ and the conditions

$$u_a \leq \eta_1 \quad \text{and} \quad (\sum_{t=1}^{T} \mu_t \Pi_t)_a \leq \eta_2 c_a. \tag{7}$$

Those two conditions turn out to be insufficient in practice to guarantee that an arc that is made inactive will not become active later on in the process. To increase our chance to have the newly made inactive arc remain inactive, we look at the contribution of the dual variable u to the linear component of the Lagrangian dual, namely the product $u_a c_a$ (see (5)). If this product is small and contributes for little to the total sum $\sum_{a \in \mathcal{A}_1} u_a c_a$, setting u_a to zero will not affect much the linear part of the Lagrangian function and plausibly not call for a serious readjustment in the second component $\mathcal{M}_{\mathcal{A}_1}(u_{\mathcal{A}_1})$. We thus have a last condition

$$u_a c_a \leq \eta_3 \sum_{a \in \mathcal{A}_1} u_a c_a, \tag{8}$$

where η_3 is a positive small enough number.

To summarize the above discussion, we have introduced two heuristic rules that move elements between $\mathcal{A}_1$ and $\mathcal{A}_2$:

Move from $\mathcal{A}_2$ to $\mathcal{A}_1$ all arcs such that $(\sum_{t=0}^{T} \mu_t \Pi_t)_a > c_a$.

Move from $\mathcal{A}_1$ to $\mathcal{A}_2$ all arcs such that $u_a \leq \eta_1$, $(\sum_{t=0}^{T} \mu_t \Pi_t)_a \leq \eta_2 c_a$ and $u_a c_a \leq \eta_3 \sum_{a \in \mathcal{A}_1} u_a c_a$.

Note that the rules assume there exists $\mu \geq 0$ feasible to (6).

5 Proximal-ACCPM to solve the dual Lagrangian

As discussed in Section 3, the dual Lagrangian (5) is a concave optimization problem. We present below a solution method for this class of problems, whose canonical representative can be written as

$$\max\{f(u) - c^T u \mid u \geq 0\}, \tag{9}$$

where f is a concave function revealed by a first order oracle. By oracle, we mean a black-box procedure that returns a support to f at the query point $\bar{u}$. This support takes the form of the *optimality cut*

$$a^T(u - \bar{u}) + f(\bar{u}) \geq f(u), \quad \text{for all } u, \tag{10}$$

where the vector $a \in R^n$ is an element of the anti-subgradient set[1] $a \in -\partial(\text{-}f(\bar{u}))$.

[1]We use the notation $\partial(\cdot)$ to designate the subgradient set of a *convex* function. In our case $-f$ is convex. An anti-subgradient of the concave function f is the opposite of a subgradient of $-f$.

The hypograph of the function f is the set $\{(z,u) \mid z \leq f(u)\}$. Problem (9) can be written in terms of the hypograph variable z as

$$\max\{z - c^T u \mid z \leq f(u),\ u \geq 0\}. \tag{11}$$

Optimality cuts (10) provide an outer polyhedral approximation of the hypograph set of f. Suppose that a certain number query points u^t, $t = 1, \ldots, T$ have been generated. The associated anti-subgradients a^t are collected in a matrix A. We further set $\gamma_t = f(u^t) - (a^t)^T u^t$. The polyhedral approximation of the hypograph 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: $\underline{\theta} = \max_t (f(u^t) - c^T u^t)$.

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

$$\mathcal{F}_{\underline{\theta}} = \{(u,z) \mid -A^T u + ze \leq \gamma,\ z - c^T u \geq \underline{\theta},\ u \geq 0\}. \tag{12}$$

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

The solution method we use is a cutting plane scheme in which each query point is chosen to be the proximal analytic center of the localization set. For the sake of clarity, let us first briefly sketch the basic step of a generic cutting plane method.

1. `Select a query point in the localization set.`
2. `Send the query point to the oracle and get back the optimality cut.`
3. `Update the lower and upper bounds and the localization set.`
4. `Test termination.`

The proximal analytic center cutting plane is defined as the unique minimizer of a logarithmic barrier for the localization set, augmented with a proximal term

$$F(u,z) = \frac{\rho}{2}||u - \underline{u}||^2 - \sum_{i=0}^{T} \log s_t - \sum_{i=0}^{n} \log u_i,$$

with $s > 0$ defined by

$$\begin{aligned} s_0 &= z - c^T u - \underline{\theta}, \\ s_t &= \gamma_t - z + (a^t)^T u, \quad t = \{1, \ldots, T\}. \end{aligned}$$

In this formula, the proximal center $\underline{u}$ is chosen as the query point u^t that achieves the best recorded value $\underline{\theta}$, i.e., $\underline{u} = \arg\max_t \{f(u^t) - c^T u^t\}$.

The proximal analytic center method defines the next query point as the u component of the solution (u,z) to the minimization problem

$$\begin{aligned} \min \quad & F(u,z) = \tfrac{\rho}{2}||u - \underline{u}||^2 - \sum_{i=0}^{T} \log s_t - \sum_{i=0}^{n} \log u_i \\ & s_0 = z - c^T u - \underline{\theta} \geq 0, \\ & s_t = \gamma_t - z + (a^t)^T u \geq 0, \quad t = \{1, \ldots, T\}. \end{aligned} \tag{13}$$

The quadratic term ensures that the augmented logarithmic barrier always has a minimum value.

In the next few paragraphs, we shall 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 (13) are

$$\begin{aligned} \rho(u-\underline{u}) - As^{-1} + s_0^{-1}c - u^{-1} &= 0, & (14)\\ e^T s^{-1} - s_0^{-1} &= 0, & (15)\\ s_0 - z + c^T u + \underline{\theta} &= 0, & (16)\\ s - \gamma + ze - A^T u &= 0. & (17) \end{aligned}$$

The algorithm that computes the analytic center is essentially a Newton method applied to (14)–(17). (A closely related method is described in [5].) To write down the formulae, we introduce the residuals

$$\begin{aligned} r_u &= -(\rho(u-\underline{u}) - As^{-1} + s_0^{-1}c - u^{-1}),\\ r_z &= -(e^T s^{-1} - s_0^{-1}),\\ r_{s_0} &= -(s_0 - z + c^T u + \underline{\theta}),\\ r_s &= -(s - \gamma + ze - A^T u). \end{aligned}$$

The Newton direction is given by

$$\begin{pmatrix} \rho I + AS^{-2}A^T + s_0^{-2}cc^T + U^{-2} & -As^{-2} - s_0^{-2}c \\ -(s^{-2})^T A^T - s_0^{-2}c^T & e^T S^{-2} e + s_0^{-2} \end{pmatrix} \begin{pmatrix} du \\ dz \end{pmatrix} = \begin{pmatrix} r_u - AS^{-2}r_s + s_0^{-2}r_{s_0}c \\ r_z + (s^{-2})^T r_s - s_0^{-2}r_{s_0} \end{pmatrix}. \quad (18)$$

Let $v = (u, z)$. The steplength $\alpha > 0$ along the search direction $dv = (du, dz)$ must ensure $u + \alpha du > 0$. In general, we take α to be a fixed fraction of $\bar{\alpha} = \arg\max\{\alpha \mid u + \alpha du > 0\}$. If $r_{s_0} = 0$ and $r_s = 0$, then the step can be determined by solving the one-dimensional problem

$$\alpha_{opt} = \arg\min_{\alpha} F(v + \alpha dv). \quad (19)$$

The Newton method can be summarized as

- `Select an initial point` v^0.
- `Basic iteration`
  1. `Compute the Newton step` dv `by (18).`
  2. `Test termination.`
  3. `Take a fixed step or perform linesearch (19) to update` v.

Upper bound on the optimal value We propose here a general technique to compute an upper bound. Since the constraints $-A^T u + ze \leq \gamma$ define a relaxation of the hypograph of the function to f, one obtains an upper bound by solving

$$\bar{\theta} = \max_{u;z}\{z - c^T u \mid -A^T u + ze \leq \gamma,\ u \geq 0\}. \quad (20)$$

or, by duality,

$$\bar{\theta} = \min_{\mu}\{\gamma^T \mu \mid A\mu \leq c,\ e^T\mu = 1,\ \mu \geq 0\}. \quad (21)$$

Assume that we have at hand a nonzero vector $\bar{\mu} \geq 0$. Without loss of generality, we assume $e^T\bar{\mu} = 1$. Let $r = c - A\bar{\mu}$. Clearly, $\bar{\mu}$ is feasible to

$$\bar{\theta} = \min_{\mu}\{\gamma^T \mu \mid A\mu \leq c + r^-,\ e^T\mu = 1,\ \mu \geq 0\}. \quad (22)$$

where $r = r^+ - r^-$ is the decomposition of r into a positive and a negative part. In view of the dual of (22)

$$\max_{u;z}\{z - (c + r^-)^T u \mid -A^T u + ze \leq \gamma,\ u \geq 0\},$$

we conclude that for all u feasible to (20)

$$z - (c + r^-)^T u \leq \gamma^T \bar{\mu}.$$

In particular,

$$f(u^*) - c^T u^* \leq \gamma^T \bar{\mu} + (r^-)^T u^*, \tag{23}$$

where u^* is an optimal solution of (9).

The quality of the bound depends on the choice of $\bar{\mu}$. We certainly wish to have the first term in the right-hand side of (23) be small. Besides, since u^* is not known, it is also desirable to have r^- as small as possible. We now propose a heuristic to choose $\bar{\mu}$ which satisfies those requirements. The heuristic uses the information collected in the computation of the analytic center. More precisely, assume that (u^c, z^c) is an approximate analytic center, in the sense that s_0^c and s^c, defined by (16) and (17), are strictly positive, and the equations (14) and (15) are approximately satisfied. Let

$$\mu^c = (s_0^c)(s^c)^{-1} > 0. \tag{24}$$

If (15) is satisfied, then $e^T \mu^c = 1$. Otherwise, we scale μ^c to meet the condition. It is easy to relate $r = c - A\mu^c$ to the residual in (14). 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. We have the chain of inequalities

$$\begin{aligned} f(u^*) - c^T u^* &\leq \gamma^T \mu^c + (r^-)^T u^*, \\ &= \gamma^T \mu^c + (r^-)^T u^c + (r^-)^T (u^* - u^c), \\ &\leq \gamma^T \mu^c + (r^-)^T u^c + ||r^-||\delta. \end{aligned} \tag{25}$$

The last inequality follows from Cauchy-Schwarz and $\delta \geq ||u^* - u^c||$ is an upper bound on the distance of the current point u^c to the optimal set. Note that if $r^- = 0$, then (25) gives an exact bound. It turns out that for the linear multicommodity flow problems the condition $r^- = 0$ is often met. Since the oracle generates true lower bounds, the difference between the two bounds provides a reliable optimality gap that is used to terminate the cutting plane algorithm.

6 The algorithm

We first review the main items in the implementation of our solution method.

Oracle To solve the shortest path problems, we use Dijkstra's algorithm [4]. This algorithm computes all the shortest paths from a single node to all other nodes in a directed graph. We partition the commodities according to the origin node of the demand and solve $|\mathcal{S}|$ Dijkstra's algorithms, one per source node, where $\mathcal{S} \subset \mathcal{N}$ is the set of all origin nodes. To improve computational time, we have implemented special data structures for the updating of the distance value of adjacent nodes and for the search of the node with the lowest distance measure. We also exploit the sparsity of the graph and the fact that the oracle solves Dijkstra's algorithm on the same graph at each iteration.

Bounds on the objective The oracle computes a solution for the dual problem by solving shortest path problems. It thus produces a lower bound $\underline{\theta}$ for the original problem.

The upper bound $\bar{\theta}$ is computed with (25). Let us interpret it in the context of linear multicommodity flow problem. Recall that a column of A can be interpreted as the result of a flow assignment that meets all demands (but not necessarily the capacity constraints). In view of the definition of the shipping cost t, we have $\gamma = A^T t$. Given a vector μ, with $e^T\mu = 1$ and $\mu \geq 0$, the vector $y = A\mu$ can be viewed as a convex combination of flow assignments. It satisfies the demand constraints, and, if $A\mu \leq c$, it also satisfies the capacity constraints. Then $\gamma^T\mu = t^T A\mu = t^T y$ is the primal cost of a fully feasible flow assignment, thus an exact upper bound. If $A\mu \not\leq c$, then $\gamma^T\mu$ is not an upper bound, but (25) still provides an approximate upper bound. We do not use it as our termination criterion. In practice, we use μ defined by (24).

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

$$(\bar{\theta} - \underline{\theta})/\max(\underline{\theta}, 1) \leq \epsilon, \tag{26}$$

where $\bar{\theta}$ is the best exact upper bound obtained so far. In our experiments we use $\epsilon = 10^{-5}$.

Proximal center The initial proximal center is the first query point. Thereafter, the proximal center is updated to the current query point whenever the oracle returns an objective function value that improves upon the best upper bound.

Proximal parameter We used a fixed value for ρ, e.g. $\rho = 10^{-2}$. For problems with high tolerance requirement, say $\epsilon = 10^{-5}$, we need not update this value. Indeed, when approaching the optimal solution, (25) keeps generating exact upper bounds. If a lower precision is required, say 10^{-3}, it may happen that (25) only generates approximate upper bounds. Instead of iterating with the same ρ until (25) delivers an exact upper bound with the required precision, we find it convenient to use the approximate upper bound to signal closeness to the solution. We then switch to $\rho = 10^{-10}$. By lowering the impact of the proximal term, it makes it easier for Proximal-ACCPM to find a primal feasible solution, i.e., an exact upper bound.

Weight on epigraph cut The localization set is bounded below by a special constraint on the objective. We named it the epigraph cut. It is easily checked that the epigraph cut makes a negative angle with the optimality cuts. When the number of optimality cuts increases, their total weight dominates the weight of the epigraph cut in (13). Thus, the epigraph cut tends to become active at the analytic center, with the possible effect of slowing the global convergence. To counteract this negative effect, we assign a weight to the epigraph cut equal to the total number of generated cuts. If we aim to a low precision, say 10^{-3}, we found that setting the weight to 50 times the total number of optimality cuts is more efficient.

Column elimination 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 μ be defined by equation (24). (We assume without loss of generality that $e^T\mu = 1$.) Then $A\mu$ is the total flow on the network. If μ_i is much smaller than the average of μ, then column i contributes little to the solution (dually, the distance s_i between the analytic center and the cut is high). Such column is a good candidate for elimination. To make the elimination test robust, we use the median of μ and eliminate the columns whose coefficient μ_i is less than $1/\kappa$ times the median. In practice, we choose $\kappa = 4$. We also perform the test once every τ iterations. A good value, is $\tau = 20$. (For the largest problem, we took $\tau = 100$.)

The algorithm is best described using a concept of a *coordinator* acting as a mediator

between the oracle that produces shortest paths and the query point generator (Proximal-ACCPM) that produces analytic centers. The coordinator keeps a repository of generated shortest paths and is responsible for managing the active set (Section 4). The coordinator also computes the relative optimality gap (26) and tests termination.

The coordinator queries the oracle at the current point u (with $u_{\mathcal{A}_2} = 0$), and transmits to Proximal-ACCPM the part of the shortest path that corresponds to the active set $\mathcal{A}_1$. It retrieves from Proximal-ACCPM a pair of primal-dual (approximate) analytic center $(u_{\mathcal{A}_1}, \mu)$. The latter is used to compute a convex combination of all shortest paths yielding an upper bound and a test for updating the active set.

The initialization phase and the basic step that compose the algorithm are described below.

Initialization

At first, all arcs are declared inactive. Solve the shortest path problems (corresponding to $u = 0$).

$$\mathcal{M} = \sum_{k \in \mathcal{K}} \min \left\{ \sum_{a \in \mathcal{A}} t_a x_a^k \mid Nx^k = \delta^k,\, x_a^k \geq 0, \quad \forall a \in \mathcal{A} \right\}.$$

Create the partition $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ using the solution x^k according to $\mathcal{A}_1 = \{a \mid \sum_{k \in \mathcal{K}} x_a^k \geq c_a\}$ and $\mathcal{A}_2 = \{a \mid \sum_{k \in \mathcal{K}} x_a^k < c_a\}$. Then, we initiate the algorithm with $u_{\mathcal{A}_1} > 0$ (in practice, take $u_a = 10^{-3}$).

Basic step

1. The oracle solves shortest path problems $\mathcal{M}_{\mathcal{A}_1}(u_{\mathcal{A}_1})$ and returns the objective function $\mathcal{L}(u_{\mathcal{A}_1})$, a new anti-subgradient vector $a \in -\partial(-\mathcal{L}(u_{\mathcal{A}_1}))$ and a path flow vector Π_j $(a = \Pi_j - c)$. This information defines a new cutting plane

$$a^T(u'_{\mathcal{A}_1} - u_{\mathcal{A}_1}) + \mathcal{L}(u_{\mathcal{A}_1}) \geq \mathcal{L}(u'_{\mathcal{A}_1}), \quad \text{for all } u'_{\mathcal{A}_1}.$$

2. Proximal-ACCPM computes a new query point $u_{\mathcal{A}_1}$, the dual variable μ (and the associated flow $y = \sum_{t=1}^{T} \mu_t \Pi_t$) and an upper bound $\bar{\theta}$ (exact or estimated).
3. Update the lower bound with

$$\underline{\theta} = \max(\underline{\theta}, \mathcal{L}(u_{\mathcal{A}_1})).$$

4. If the termination criterion (relative optimality gap criterion with a primal feasible solution) is reached, then STOP
5. Update the partition $\mathcal{A} = \mathcal{A}_1 \cup \mathcal{A}_2$ using the flow vector y and the dual variables $u_{\mathcal{A}_1}$.
   - All inactive arcs $a \in \mathcal{A}_2$ such that $y_a \geq c_a$ are declared active.
   - All active arcs $a \in \mathcal{A}_1$ such that $u_a \leq \eta_1$, $y_a \leq \eta_2 c_a$ and

$$u_a c_a \leq \eta_3 \sum_{a \in \mathcal{A}_1} u_a c_a,$$

   are declared inactive.

7 Numerical results

7.1 Test problems

We used four sets of test problems. The first set, the `planar` problems, contains 10 instances and has 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. Arc costs are Euclidean distances, while 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. The arc costs, commodities, and demands are generated in a way similar to that of `planar` networks. These two sets of problems are solved in [11]. The data can be downloaded from `http://www.di.unipi.it/di/groups/optimize/Data/MMCF.html`.

The third collection of problems is composed of telecommunication problems of various sizes. The cost functions for these problems are originally non-linear. To make them linear, we use different techniques depending on the type of non-linear cost function that was used. In the small `ndo22` and `ndo148` problems, the cost functions have a vertical asymptote. We use this asymptote as a natural capacity bound. Problem `904` is based on a real telecommunication network. It has 904 arcs and 11130 commodities and was used in the survey paper [16].

The last collection of problems is composed of transportation problems. The problems `Sioux-Falls`, `Winnipeg`, `Barcelona` are solved in [11]; there the demands of `Winnipeg` and `Barcelona` are divided, as in [11], by 2.7 and 3 respectively, to make those problems feasible. The last three problems, `Chicago-sketch`, `Chicago-region` and `Philadelphia` can be downloaded from `http://www.bgu.ac.il/~bargera/tntp/`. The data include an increasing congestion function that is not adapted to our formulation. This function uses capacity and "free flow time". We use this free flow time as unit cost. To turn those problems into linear ones we use the following strategy. For each problem, we divide all the demands by a same coefficient. We increase this coefficient until the problem becomes feasible with respect to the capacity constraints. We end up using coefficients 2.5, 6 and 7 for problems `Chicago-sketch`, `Chicago-region` and `Philadelphia`, respectively.

Table 1 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 cost value z^* of an optimal solution to (1) with a relative optimality gap less than 10^{-5}. Some instances are huge. In Section 2 we mentioned that the most compact formulation of the linear programming problem as a single linear program involves $|\mathcal{N}| \times |\mathcal{A}|$ variables and $|\mathcal{A}| + |\mathcal{N}|^2$ variables. This means over 500 millions variables and nearly 200 millions constraints for problem `Philadelphia`. The last column of Table 1 displays the percentage of saturated arcs, denoted $\%\frac{|\mathcal{A}_1^*|}{|\mathcal{A}|}$, at the optimum. Note that the figures in the last column are low, in particular for the real-life transportation problems.

7.2 Numerical experiments

The goals of the numerical study are five-fold. First, we compare Proximal-ACCPM with the standard ACCPM. The later method has no proximal term but uses artificial bounds on the dual variables u to ensure compactness of the localization set. In this comparison we do not use the active set strategy. In the second set of experiments, we analyze the impact of column elimination while in the third one, we focus on the active set strategy. In the forth experiments, we combine column elimination and the active set strategy to achieve the fastest computing time. Finally, in the last experiment, we compare our solution method with the augmented Lagrangian algorithm of T. Larsson and Di Yuan [11].

Problem ID	$\lvert\mathcal{N}\rvert$	$\lvert\mathcal{A}\rvert$	$\lvert\mathcal{K}\rvert$	z^*	$\%\frac{\lvert\mathcal{A}_1^*\rvert}{\lvert\mathcal{A}\rvert}$
`planar` problems					
`planar30`	30	150	92	4.43508×10^7	9.3
`planar50`	50	250	267	1.22200×10^8	11.6
`planar80`	80	440	543	1.82438×10^8	24.3
`planar100`	100	532	1085	2.31340×10^8	16.3
`planar150`	150	850	2239	5.48089×10^8	27.5
`planar300`	300	1680	3584	6.89982×10^8	7.4
`planar500`	500	2842	3525	4.81984×10^8	2.0
`planar800`	800	4388	12756	1.16737×10^8	3.0
`planar1000`	1000	5200	20026	3.44962×10^9	9.6
`planar2500`	2500	12990	81430	1.26624×10^{10}	14.7
`grid` problems					
`grid1`	25	80	50	8.27323×10^5	8.7
`grid2`	25	80	100	1.70538×10^6	25.0
`grid3`	100	360	50	1.52464×10^6	4.2
`grid4`	100	360	100	3.03170×10^6	8.3
`grid5`	225	840	100	5.04970×10^6	3.7
`grid6`	225	840	200	1.04007×10^7	13.5
`grid7`	400	1520	400	2.58641×10^7	7.0
`grid8`	625	2400	500	4.17113×10^7	11.8
`grid9`	625	2400	1000	8.26533×10^7	16.3
`grid10`	625	2400	2000	1.64111×10^8	16.3
`grid11`	625	2400	3000	3.29259×10^8	11.0
`grid12`	900	3480	6000	5.77189×10^8	6.2
`grid13`	900	3480	12000	1.15932×10^9	8.0
`grid14`	1225	4760	16000	1.80268×10^9	3.5
`grid15`	1225	4760	32000	3.59353×10^9	4.0
Telecommunication-like problems					
`ndo22`	14	22	23	1.88237×10^3	9.0
`ndo148`	58	148	122	1.39500×10^5	0
`904`	106	904	11130	1.37850×10^7	9.2
Transportation problems					
`Sioux-Falls`	24	76	528	3.20184×10^5	2.6
`Winnipeg`	1067	2975	4345	2.94065×10^7	2.0
`Barcelona`	1020	2522	7922	3.89400×10^7	0.4
`Chicago-sketch`	933	2950	93513	5.49053×10^6	1.0
`Chicago-region`	12982	39018	2297945	3.06541×10^6	0.6
`Philadelphia`	13389	40003	1151166	1.65428×10^7	0.4

Table 1: Test problems: optimal value with 10^{-5} optimality gap.

For all results using Proximal-ACCPM and 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 Proximal-ACCPM is reduced to the active arcs only. Thus, we also give the percentage of arcs in the active set, $\%\frac{|\mathcal{A}_1|}{|\mathcal{A}|}$, and the percentage of saturated arcs, $\%\frac{|\mathcal{A}_1^*|}{|\mathcal{A}|}$, at the end of the solution process.

The Proximal-ACCPM code we use has been developed in Matlab at the Logilab laboratory, 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.

7.2.1 Proximal-ACCPM vs. ACCPM

In this subsection Proximal-ACCPM and ACCPM are implemented without using the active set strategy. The ACCPM code we use is essentially the same as the Proximal-ACCPM code in which we just set the proximal parameter to zero and introduce instead upper bounds on the variables to enforce compactness in the initial phase. In our experiments, the default upper bounds are chosen to be quite large, say 10^6, to be inactive at the optimum. All problems, but three, are solved with a relative gap of 10^{-5}. None of the two algorithms could solve `planar2500`, `Chicago-region` and `Philadelphia`, partly because too many cuts in the localization set jammed the memory space. Table 2 shows that the results with Proximal-ACCPM and ACCPM are quite similar. The number of basic steps and the computational times are more or less the same. For this class of problems, the two methods appear to be equivalent. However, we will use Proximal-ACCPM in all further experiments, because it is easier to manipulate the unique parameter ρ than implementing individual strategies to move the upper bounds on the variables.

7.2.2 Column elimination

In this subsection, Proximal-ACCPM solves the set of problems using column elimination. We report the results in Table 3. Column *Nb cuts* displays the number of remaining cuts at the end of the process while the last column *CPU Ratio* gives the improvement ratio of the CPU time of Proximal-ACCPM without using column elimination strategy (see Table 2), and with column elimination strategy. In this table all problems are solved with a relative optimality gap of 10^{-5}, except `planar2500` that is solved with a 10^{-4} precision. We observe a speed-up 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 Proximal-ACCPM. It is apparent in comparing the proportion of time spent in the oracle (column 5 in Table 2 and column 6 in Table 3).

7.2.3 Active set strategy

In this subsection, Proximal-ACCPM solves the linear multicommodity flow problems with the active set strategy. Table 4 shows the results with a relative optimality gap of 10^{-5}. `planar2500` is solved with a relative gap of 1.2×10^{-5}. In the last column of the table, we give the improvement ratio of the CPU time of Proximal-ACCPM without using active set strategy (see Table 2), and with the active set strategy (see Table 4).

Table 4 shows that the active set strategy reduces the CPU time with a factor around 2 to 8 on most problems. The reduction is achieved in the computation of the analytic center. Indeed, the complexity of an inner iteration is roughly proportional to the square of the dimension of the working space. As shown in the second column of Table 4, the dimension of the working space —measured by $|\mathcal{A}_1|$ at the end of the iterations— is a small fraction of the total number of arcs. It is also interesting to note that the active

	Proximal-ACCPM				**ACCPM**			
Problem ID	Outer	Inner	CPU	%Or	Outer	Inner	CPU	%Or
planar30	59	176	0.7	21	59	189	0.8	18
planar50	109	266	1.9	20	106	286	2.1	18
planar80	281	617	20.5	9	277	645	19.8	9
planar100	263	593	20.4	9	265	638	19.4	10
planar150	688	1439	330.0	2	700	1544	339.6	2
planar300	374	909	122.2	2	384	988	129.6	2
planar500	229	744	88.7	21	231	799	88.6	21
planar800	415	1182	557.2	16	419	1270	553.9	17
planar1000	1303	2817	7846.7	12	1314	2995	7896.8	12
planar2500	-	-	-	-	-	-	-	-
grid1	35	114	0.3	26	36	120	0.4	17
grid2	73	222	0.8	30	77	251	1.0	26
grid3	65	239	1.2	21	66	261	1.5	18
grid4	99	319	2.4	21	97	326	2.5	19
grid5	121	414	7.3	21	120	420	7.6	20
grid6	315	770	45.1	11	313	815	45.3	11
grid7	308	827	80.0	16	317	901	87.0	15
grid8	686	1601	893.9	8	691	1686	812.0	8
grid9	942	2082	1793.8	6	942	2159	1798.1	6
grid10	946	2096	1885.1	6	947	2173	1842.6	6
grid11	648	1515	715.1	10	647	1565	702.3	11
grid12	509	1341	658.5	18	507	1397	644.8	18
grid13	673	1629	1226.8	12	679	1728	1214.4	12
grid14	462	1363	843.6	22	469	1450	845.4	22
grid15	520	1450	1055.1	20	522	1529	1045.6	20
ndo22	18	59	0.1	12	18	62	0.1	12
ndo148	17	82	0.2	20	17	83	0.3	18
904	269	640	33.2	12	271	671	34.7	12
Sioux-Falls	30	95	0.3	24	32	117	0.4	19
Winnipeg	224	592	81.2	18	258	942	120.8	14
Barcelona	157	421	35.9	23	156	457	37.8	22
Chicago-sketch	180	493	79.2	47	182	523	83.6	45
Chicago-region	-	-	-	-	-	-	-	-
Philadelphia	-	-	-	-	-	-	-	-

Table 2: Proximal-ACCPM vs. ACCPM.

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	CPU ratio
planar30	40	63	142	0.7	18	1.0
planar50	68	104	224	1.8	27	1.1
planar80	145	274	593	14.5	19	1.4
planar100	105	338	705	14.2	23	1.4
planar150	314	814	1641	136.9	16	2.4
planar300	171	383	803	80.7	25	1.5
planar500	103	221	495	59.4	37	1.5
planar800	188	388	845	281.3	39	2.0
planar1000	645	1058	2148	2861.4	20	2.7
planar2500*	1628	2156	4349	47355.8	18	-
grid1	31	35	97	0.3	18	1.0
grid2	46	60	132	0.6	19	1.3
grid3	47	63	193	1.2	17	1.0
grid4	60	93	249	2.0	18	1.2
grid5	81	123	364	6.1	20	1.2
grid6	164	308	683	28.5	21	1.6
grid7	182	312	749	55.2	23	1.4
grid8	385	706	1526	503.8	16	1.8
grid9	538	959	2008	1039.6	15	1.7
grid10	532	969	2022	1043.8	16	1.8
grid11	350	663	1453	434.6	22	1.6
grid12	280	520	1258	436.2	30	1.5
grid13	362	687	1575	773.5	25	1.6
grid14	231	478	1296	539.7	39	1.6
grid15	275	537	1367	696.9	36	1.5
ndo22	18	18	59	0.1	12	1.0
ndo148	17	17	82	0.2	20	1.0
904	119	254	533	19.1	21	1.7
Sioux-Falls	24	30	93	0.3	23	1.0
Winnipeg	118	227	613	54.5	25	1.5
Barcelona	105	156	438	28.9	27	1.2
Chicago-sketch	108	178	491	55.1	41	1.4
Chicago-region	460	1376	3030	44117.3	64	-
Philadelphia	326	885	1859	22937.2	66	-

* Problem solved with a relative optimality gap of 10^{-4}.

Table 3: Proximal-ACCPM with column elimination.

Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert\mathcal{A}\rvert}$	$\%\frac{\lvert\mathcal{A}_1^*\rvert}{\lvert\mathcal{A}\rvert}$	Outer	Inner	CPU	%Or	CPU ratio
planar30	12.7	9.3	46	154	0.5	24	1.5
planar50	13.2	11.6	99	258	1.1	32	1.7
planar80	25.7	24.3	279	656	7.4	24	2.8
planar100	17.5	16.3	260	626	5.9	31	3.5
planar150	29.0	27.5	716	1597	93.8	8	3.5
planar300	9.1	7.4	343	835	15.4	18	7.9
planar500	2.6	2.0	140	359	12.8	87	6.9
planar800	3.6	3.0	317	786	81.7	85	6.8
planar1000	10.4	9.6	1249	2860	1244.9	36	6.3
planar2500*	15.8	14.7	2643	7160	34022.2	21	-
grid1	12.5	8.7	24	83	0.2	25	1.4
grid2	32.5	25.0	61	202	0.7	30	1.2
grid3	5.0	4.2	37	104	0.3	46	3.8
grid4	10.3	8.3	79	213	1.0	39	2.4
grid5	6.0	3.7	90	256	1.9	60	3.9
grid6	20.6	13.5	294	731	13.3	35	3.4
grid7	9.3	7.0	264	704	19.1	58	4.2
grid8	13.2	11.8	623	1465	155.5	37	5.1
grid9	18.1	16.3	907	2158	413.6	25	4.3
grid10	18.0	16.3	919	2209	432.7	26	4.4
grid11	12.2	11.0	569	1391	140.3	46	5.1
grid12	7.4	6.2	394	979	121.4	74	5.4
grid13	9.6	8.0	558	1333	209.4	59	5.9
grid14	4.4	3.5	310	767	139.8	89	6.0
grid15	4.9	4.0	364	902	173.7	86	6.1
ndo22	9.0	9.0	11	49	0.1	11	1.5
ndo148	1.7	0.0	2	13	0.01	35	14.6
904	13.1	9.2	321	984	15.9	30	2.1
Sioux-Falls	7.9	2.6	13	56	0.1	33	3.2
Winnipeg	2.9	2.0	158	393	12.8	83	6.3
Barcelona	0.5	0.4	35	111	2.1	86	17.0
Chicago-sketch	1.2	1.0	60	195	13.0	95	6.1
Chicago-region	1.1	0.6	683	1961	14684.2	98	-
Philadelphia	0.6	0.4	193	529	3125.2	99	-

* Problem solved with a relative optimality gap of 1.2×10^{-5}.

Table 4: Proximal-ACCPM with the active set strategy.

set strategy leads to a satisfactory estimate of the set of saturated arcs at the optimum. Table 4 shows that the percentage of arcs in the active set $\%\frac{|\mathcal{A}_1|}{|\mathcal{A}|}$ and the percentage of saturated arcs $\%\frac{|\mathcal{A}_1^*|}{|\mathcal{A}|}$ are very close. Last but not least, the active set strategy has a favorable, though nonintuitive, influence on the total number of outer iterations.

7.2.4 Active set strategy with column elimination

In this set of experiments, we combine column elimination and the active set strategy. The results are displayed on Table 5. Column *Nb cuts* displays the number of remaining cuts at the end of the process. The last two columns display CPU ratios. The next to the last column gives the ratio between the CPU times in Table 4 and Table 5, while the last column does a similar comparison between Table 2 and Table 5. As expected, column elimination is efficient, not only because it decreases the time spent in computing analytic center, but also because it often permits a reduction of the total number of outer iterations. This last observation is rather surprising.

Problem ID	Nb cuts	Outer	Inner	CPU	%Or	CPU ratios	
planar30	30	48	150	0.4	29	1.1	1.7
planar50	61	97	252	1.1	32	1.0	1.8
planar80	144	283	703	6.5	28	1.1	3.1
planar100	110	257	641	5.2	34	1.1	3.9
planar150	296	820	1893	64.5	13	1.5	5.1
planar300	199	325	721	9.7	27	1.6	12.6
planar500	76	118	258	10.5	90	1.2	8.5
planar800	168	252	545	60.7	91	1.3	9.2
planar1000	628	890	1904	572.6	55	2.2	13.7
planar2500	2089	3009	7546	29457.3	28	-	-
grid1	24	24	83	0.2	25	1.0	1.4
grid2	41	52	164	0.6	31	1.2	1.5
grid3	32	32	88	0.3	39	1.0	3.7
grid4	47	66	175	0.7	46	1.4	3.3
grid5	58	75	194	1.6	58	1.2	4.5
grid6	165	239	607	8.3	46	1.6	5.4
grid7	159	228	582	13.7	70	1.4	5.8
grid8	374	528	1248	97.5	50	1.6	8.1
grid9	518	720	1680	212.8	38	1.9	8.4
grid10	499	722	1654	215.6	41	2.0	8.7
grid11	329	458	1094	84.9	62	1.7	8.4
grid12	232	329	803	88.9	84	1.4	7.4
grid13	343	460	1086	136.8	74	1.5	9.0
grid14	171	252	623	107.0	94	1.3	7.9
grid15	206	294	708	131.7	91	1.3	8.0
ndo22	11	11	49	0.1	11	1.0	1.5
ndo148	2	2	13	0.01	35	1.0	14.6
904	171	311	819	12.2	38	1.3	2.7
Sioux-Falls	2.6	13	56	0.1	33	1.0	3.2
Winnipeg	93	143	372	11.1	87	1.2	7.3
Barcelona	35	35	111	2.1	86	1.0	17.0
Chicago-sketch	44	65	170	12.9	96	1.0	6.1
Chicago-region	524	742	2080	15012.1	99	1.0	-
Philadelphia	127	192	525	3092.3	99	1.0	-

Table 5: Proximal-ACCPM with active set strategy and column elimination.

7.2.5 Proximal-ACCPM vs. ALA

In [11], the authors propose an augmented Lagrangian algorithm (ALA) to generate feasible solutions with a reasonable precision. In this section, we compare Proximal-ACCPM to their algorithm. In [11], the authors introduce the concept of "near optimal solution" to designate feasible solutions with a relative optimality gap around 10^{-3} (actually, ranging from 6.10^{-4} to 6.10^{-3}). Their solution method consists in running twice their algorithm, a first pass to compute a lower bound and a second pass to compute an upper bound. The total CPU time that is reported in Table 6, is the sum of the two CPU times. To make a valid comparison, we aimed to results with a similar precision. Since the precision is moderate, we had to resort to the strategy defined in Section 6.

Table 6 displays the comparative results on the instances used in [11]. We report original computing times. Since the machines are different (a Pentium IV, 2.8 GHz with 2 Gb of RAM and a Sun ULTRASparc with 200 MHz processor and 2 GB of physical RAM), we used an artifact to estimate the speed ratio. We solved a large set of problems on the Pentium IV and on a Sun ULTRASparc with 500 MHz processor, the only Sun ULTRASparc we have at our disposal. We found a ratio of 4 between the Pentium IV and the Sun ULTRASparc 500, and we propose a 2.5 ratio between the two SUN's. Finally, we retain a factor of 10 between our machine and the one used in [11]. The last column of Table 6 gives a CPU ratio between the CPU times of Proximal-ACCPM and the CPU times of ALA. This CPU ratio includes the speed ratio between the two computers. Of course, those ratios are just indicative.

Table 6 shows that ALA is more efficient on the smaller instances[2] while the reverse holds for the larger ones[3]. In view of the active set strategy discussed in the subsection (7.2.3), we propose the following explanation for the behavior of Proximal-ACCPM on small problems. Note that the percentage of time spent in the oracle vs. the master program steadily increases with the problem dimension. This suggests a possible computing overhead in Proximal-ACCPM. Indeed, Proximal-ACCPM is written in Matlab, while the oracle is implemented in C. Moreover, Proximal-ACCPM is a general purpose code that is designed to handle a very large variety of problems. Consequently, the code contains a lot of structures that are costly to manipulate. However, on the large instances, the linear algebra operations dominate and Matlab is very efficient in performing them. An implementation of Proximal-ACCPM in C would presumably improve the performance, essentially on the smaller instances.

Table 7 displays the results for the other instances that are not considered in [11]. We solve them with a precision of 10^{-3}. For the two larger problems (`Chicago-region` and `Philadelphia`) we also give the results to obtain the first feasible solution without any condition on the relative gap. The reader will observe that our solution method produces a feasible solution for the larger problems in a very short time and with a reasonable relative optimality gap (around 10^{-2}). The computing time to gain one digit of accuracy is important. Yet, the overall time with a 10^{-3} relative precision is moderate.

Conclusion

In the present study, we used a special version of the proximal analytic center cutting plane method to solve the linear multicommodity flow problem. The main new feature is the use of an active set strategy: it cuts down computational times by a factor from 2 to 14 and permits to solve the three larger instances that could not be solved previously. We also test the ability of our method to produce fast "near optimal solutions" in the sense of [11]. In that paper, the authors used an augmented Lagrangian algorithm (ALA). We

[2]Problems smaller than `planar150` and `grid9` and also `Sioux-Falls`.

[3]Problems larger than `planar150` and `grid9` and `Winnipeg` and `Barcelona`.

	Proximal-ACCPM						ALA		Ratio
Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert\mathcal{A}\rvert}$	Outer	Inner	CPU	%Or	Gap	CPU	Gap	CPU
planar30	11.3	35	178	0.4	21	0.0014	0.18	0.002	0.04
planar50	12.8	52	304	0.9	21	0.0023	1.28	0.0032	0.15
planar80	25.9	101	487	2.6	25	0.0045	12.25	0.0019	0.47
planar100	17.8	82	396	1.8	32	0.0026	12.51	0.0021	0.69
planar150	30	187	925	13.4	14	0.0044	61.80	0.0026	0.46
planar300	8.8	56	280	1.5	31	0.0018	103.09	0.0016	**7**
planar500	2.3	27	116	3.5	87	0.0010	211.22	0.0017	**8.5**
planar800	3.5	41	184	9.8	92	0.0014	1572.33	0.0017	**16**
planar1000	11.1	108	450	45.2	85	0.0021	3097.22	0.0018	**6.8**
planar2500	15.6	229	896	707.0	87	0.0018	34123.14	0.0013	**4.8**
grid1	11.1	17	104	0.17	20	0.0007	0.040	0.0062	0.02
grid2	28.7	26	161	0.42	21	0.0015	0.12	0.0060	0.03
grid3	4.4	12	63	0.17	29	0.0010	0.44	0.0022	0.26
grid4	9.2	23	106	0.35	34	0.0012	0.23	0.0029	0.07
grid5	6	20	115	3.8	7	0.0009	1.31	0.0012	0.03
grid6	15.6	35	190	1.1	51	0.0016	2.28	0.0023	0.2
grid7	11.9	23	151	1.4	70	0.0014	8.10	0.0016	0.6
grid8	17.2	42	219	4.9	78	0.0017	19.94	0.0020	0.4
grid9	19.5	50	261	7.4	76	0.0016	52.17	0.0023	0.7
grid10	22.9	48	233	7.3	80	0.0017	104.41	0.0022	**1.4**
grid11	16.7	34	189	4.7	82	0.0015	262.54	0.0014	**5.6**
grid12	13	22	140	5.6	90	0.0014	248.57	0.0019	**4.4**
grid13	15.6	26	157	6.5	89	0.0014	948.26	0.0019	**14.7**
grid14	7.8	16	102	6.7	95	0.0012	1284.79	0.0016	**19**
grid15	8.5	21	136	9.0	95	0.0012	2835.03	0.0014	**31.2**
Sioux-Falls	7.9	8	33	0.1	23	0.0023	0.47	0.0043	0.5
Winnipeg	3.4	65	203	5.2	84	0.00055	239.20	0.00061	**4.6**
Barcelona	1.6	30	85	1.9	82	0.00034	283.64	0.00057	**15**

Table 6: Proximal-ACCPM vs. ALA.

Problem ID	$\%\frac{\lvert\mathcal{A}_1\rvert}{\lvert\mathcal{A}\rvert}$	Outer	Inner	CPU	%Or	Gap
ndo22	9	7	22	0.06	12	10^{-3}
ndo148	-	2	13	0.01	40	10^{-9}
904	14.5	248	721	10.6	35	10^{-3}
Chicago-sketch	1.2	12	50	2.6	96	10^{-3}
Chicago-region	2.9	12	274	213.8	99	0.026
Chicago-region	2.3	111	463	2239.8	99	10^{-3}
Philadelphia	1.2	8	145	112.5	99	0.012
Philadelphia	1.1	40	203	619.4	99	10^{-3}

Table 7: Proximal-ACCPM.

compared our results to theirs and observed an acceleration factor from 2 to 31, on the large problem instances. We did not perform a comparative study with other methods, e.g., the Dantzig-Wolfe algorithm and the Bundle method, but we have noted in [11] that ALA is competitive with those two algorithms.

Let us review some directions for future research. First, we believe that the method can be accelerated by exploiting the fact that the objective is the sum of independent components. In previous studies on the nonlinear multicommodity flow problem [8, 16], the implementation exploited the fact that the function f in (9) is the sum of $|\mathcal{K}|$ independent functions. It associates with each one of them an epigraph variable and optimality cuts. A much richer information is thus transferred to the master program that enables convergence in very few outer iterations (often less than 15 on large problems). In the meantime, the computation time of the analytic centers dramatically increases. As a result, 95% of the time is spent in the master program that computes analytic centers [8, 16]. In the present paper, the proportion is just reverse: we observe that 95% of the time is spent in solving shortest path problems for the larger problem instances. If we could achieve a better balance between the two components of the algorithm, we would improve performance.

Second, it will be interesting to see whether our method performs well on other formulations of the LMCF. There exist at least two different versions of LMCF. In the formulation used in this paper, the flows of all commodities compete equally for the capacity on the arcs. In other formulations, often motivated by road traffic considerations, the commodities belong to families. The unit shipment cost on an arc depends on the commodity, and all commodities can compete for the mutual or/and individual capacity. For each family of commodities, flows may be further restricted to a subnetwork, see e.g. [1, 7]. In the Lagrangian relaxation, the subproblems are min cost flow problems that must be solved by an algorithm that cannot reduce to the shortest path. Since the two formulations yield different implementations the papers in the literature deal with either one of them, but not with both simultaneously.

References

[1] J. Castro. A specialized interior-point algorithm for multicommodity network flows. *SIAM journal on Optimization*, 10 (3):852–877, 2000.

[2] P. Chardaire and A. Lisser. Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems. *Operations Research*, 50(2):260–276, 2002.

[3] G.B. Dantzig and P. Wolfe. The decomposition algorithm for linear programming. *Econometrica*, 29(4):767–778, 1961.

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

[5] O. du Merle and J.-P. Vial. Proximal ACCPM, a cutting plane method for column generation and Lagrangian relaxation: application to the p-median problem. Technical report, Logilab, University of Geneva, 40 Bd du Pont d'Arve, CH-1211 Geneva, Switzerland, 2002.

[6] J.M. Farvolden, W.B. Powell, and I.J. Lustig. A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem. *Operations Research*, 41(4):669–693, 1993.

[7] A. Frangioni and G. Gallo. A bundle type dual-ascent approach to linear multicommodity min-cost flow problems. *Informs Journal on Computing*, 11(4):370–393, 1999.

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

[9] J.-L. Goffin, A. Haurie, and J.-P. Vial. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Science*, 38(2):284–302, 1992.

[10] J.E. Kelley. The cutting plane method for solving convex programs. *Journal of the SIAM*, 8:703–712, 1960.

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

[12] C. Lemaréchal. Bundle methods in nonsmooth optimization. *IIASA Proceedings series*, 3:79–102, 1977. C. Lemaréchal and R. Mifflin eds.

[13] J.W. Mamer and R.D. McBride. A decomposition-based pricing procedure for large-scale linear program : An application to the linear multicommodity flow problem. *Management Science*, 46:693-709, 2000.

[14] R.D. McBride and J.W. Mamer. Solving the undirected multicommodity flow problem using a shortest path-based pricing algorithm. *Networks*, 38(4):181-188, 2001.

[15] R.D. McBride. Progress made in solving the multicommodity flow problem *SIAM journal on Optimization*, 8(4):947-955, 1998.

[16] A. Ouorou, P. Mahey, and J.-P. Vial. A survey of algorithms for convex multicommodity flow problems. *Management Science*, 46:126-147, 2000.