A Survey of Algorithms for Convex Multicommodity Flow Problems

A. Ouorou • P. Mahey • J.-Ph. Vial
LIMOS—Université Blaise Pascal, ISIMA, BP 125, 63173 Aubière Cedex, France
LIMOS—Université Blaise Pascal, ISIMA, BP 125, 63173 Aubière Cedex, France
HEC, Section of Management Studies, University of Geneva, 102, Bd. Carl-Vogt, 1211 Geneva 4, Switzerland
adam.ouorou@cnet.francetelecom.fr • philippe.mahey@isima.fr • jean-philippe.vial@hec.unige.ch

Routing problems appear frequently when dealing with the operation of communication or transportation networks. Among them, the message routing problem plays a determinant role in the optimization of network performance. Much of the motivation for this work comes from this problem which is shown to belong to the class of nonlinear convex multicommodity flow problems. This paper emphasizes the message routing problem in data networks, but it includes a broader literature overview of convex multicommodity flow problems. We present and discuss the main solution techniques proposed for solving this class of large-scale convex optimization problems. We conduct some numerical experiments on the message routing problem with four different techniques.
(*Network Optimization; Multicommodity Flows; Message Routing; Convex Programming*)

1. Introduction

The literature dealing with multicommodity flow problems is important since the publication of the works of Ford and Fulkerson (1962) and Hu (1963) in the beginning of the 1960s. These problems usually have a very large number of variables and constraints and arise in a great variety of applications. The linear multicommodity flow problems have naturally attracted the interest of researchers in operations research, first as basic optimization models for network design and operation problems, and then as a natural illustration of decomposable structures in linear optimization (Gondran and Minoux 1979, Ahuja et al. 1993). Excellent surveys such as Kennington's (1978) and Assad's (1978) were published at the end of the 1970s. Numerical experiments with subgradient methods have been conducted with noticeable successes. Those methods converge slowly but seem to be relatively insensitive to the number of dual variables associated with the relaxation of the coupling constraints (Gondran and Minoux 1979). More recently, the availability of multiprocessor and massively parallel architectures have increased the interest for decomposition techniques, inducing efficient versions of known decomposition algorithms for linear programming such as Dantzig-Wolfe's algorithm (see for example Jones et al. 1993) partitioning (see for example Farvolden et al. 1993), which are able to solve large-scale network flow problems.

The first models with nonlinear costs appeared later and were studied in connection with telecommunications and transportation networks. For instance, one of the most important problems in the design of packet-switched computer networks consists of determining routes on which packets have to be transmitted to achieve optimality according to some chosen cost criterion. The average message delay is the most frequently used performance measure in the literature for such networks. Under appropriate assumptions, this problem belongs to the class of nonlinear convex multicommodity flow problems (Fratta et al. 1973, Bertsekas and Gallager 1987). Much of the motivation

0025-1909/00/4601/0126$05.00
1526-5501 electronic ISSN

for this work comes from these message routing problems.

The present study is concerned only with the nonlinear convex models for which, to our knowledge, no overview has been published. In theory these models can be solved by general nonlinear programming techniques. However, their special structure makes decomposition methods much more attractive. That is, much efficiency can be gained by identifying easier subproblems for which polynomial algorithms can be coded very efficiently, such as shortest path and minimum cost flow calculations.

After reviewing the existing literature, we have chosen to focus on four main algorithms: the Flow Deviation Method (FD), the Projected Newton Method (PM), the Analytic Center Cutting Plane Method (ACCPM), and the Proximal Decomposition Method (PDM). We briefly describe their underlying ideas and perform a comparative analysis of their performance on a real-life problem. We did not submit the other methods to numerical testing, partly because the codes are not readily available, and partly because the nontested methods can be viewed as variants or enhancements of the methods we analyze more in depth. The Flow Deviation algorithm (LeBlanc 1973, Fratta et al. 1973) is the oldest method among these four algorithms and it is still the most popular in both transportation or telecommunication models. The reason is twofold: The method easily generates feasible solutions from shortest path calculations and enjoys fast convergence in the early iterations. The projection algorithm proposed by Bertsekas et al. (1984), and Bertsekas and Gallager (1987) is a primal method too. It alternates shortest path calculations with projected Newton steps, allowing faster convergence when twice differentiable objective functions are at hand. The other two algorithms are less classical dual approaches. ACCPM (Goffin et al. 1997) belongs to the family of cutting plane methods and PDM (Mahey et al. 1998) to distributed algorithms issued from the Proximal Point Algorithm (Rockafellar 1976).

Work has been done in the direction of nonconvex costs. See, for instance, the survey of Minoux (1989) for multicommodity flow problems with concave differentiable cost functions and/or fixed cost.

2. The Convex Multicommodity Flow Problem

We use the following notation. We are given a directed graph $G = (V, E)$ associated with a network with m nodes (which according to the context may represent switching centers, terminals, concentrators, and so on) and n arcs (links between some chosen pairs of nodes). Let K denote the number of commodities to be transported through the network. Each commodity k has a single-source sink pair (s_k, t_k) and we are given the flow requirement r_k (traffic quantity which must be sent between s_k and t_k). The general nonlinear convex multicommodity flow problems we are concerned with may be formulated as:

$$\min f(x) = \sum_{j=1}^{n} f_j(x_{0j}) + \sum_{k=1}^{K} \sum_{j=1}^{n} f_{kj}(x_{kj})$$

$$\text{s.t. } x_{0j} = \sum_{k=1}^{K} x_{kj}, \quad j = 1, \ldots, n, \tag{1}$$

$$Mx_k = r_k b_k, \quad k = 1, \ldots, K, \tag{2}$$

$$0 \le x_k, \quad k = 1, \ldots, K, \tag{3}$$

$$0 \le x_{0j} \le c_j, \quad j = 1, \ldots, n, \quad \text{where} \tag{4}$$

M	is the $m \times n$ node-arc incidence matrix of G,
x_k	denotes the n-vector representing the kth commodity,
b_k	is the m-vector with all components 0 except $b_{ks_k} = -b_{kt_k} = 1$,
c_j	is the capacity of arc j, and
x_{0j}	represents the total flow on arc j.

This model is known as a node-arc formulation. Constraints (2) and (4) are respectively classical network and capacity constraints. Constraints (1) are coupling constraints in the sense that, when relaxed, K independent individual flow problems can be solved separately. The nonnegativity constraints (3) express the fact that arcs are used in the direct way. The functions f_{kj} are associated with the flow of each commodity on each arc of the network and are supposed to be convex, as well as the functions f_j associated with the total flow x_{0j}.[1]

The model uses directed graphs. In some applications encountered in practice, flows on the edges of the network are bidirectional. In this case, each edge $j = \{u, v\}$ is substituted by two directed arcs $j^+ = (u, v)$ and $j^- = (v, u)$ in both directions. The node-arc incidence matrix is then defined for the digraph $G' = (V, U)$ where U contains the arcs j^+ and j^- for each edge j, and the cost function f_j is charged on the total edge flow x_{0j} given by (see for example, Goffin et al. 1997)

$$x_{0j} = \sum_{k=1}^{K} (x_{kj^+} + x_{kj^-}), \quad x_{kj^+}, x_{kj^-} \geq 0.$$

For sake of simplicity, we will not include the case of undirected graphs in the sequel.

Among the problems related to routing in data networks, the *message routing problem* plays an important role in the optimization of network performance. This problem consists in the determination of the sets of routes on which packets have to be transmitted in order to optimize some cost function, which measures the global quality of service when operating the network. Most models (see Kleinrock 1964, Bertsekas and Gallager 1987) use the Kleinrock average delay function, i.e., the sum over all arcs of the average delay proportional to

$$f_j(x_{0j}) = \frac{x_{0j}}{c_j - x_{0j}} \tag{5}$$

which imposes $x_{0j} < c_j$ in (4). In this application, $f_{kj} \equiv 0$.

The above multicommodity flow problem can alternatively be formulated using flows through paths of the network. More precisely, let N_k denote the number of paths between the nodes s_k and t_k, and π_{kp} the arc-path incidence vector of the pth path defined by

$$\pi_{kp}(j) = \begin{cases} 1 & \text{if } j \text{ belongs to the path,} \\ 0 & \text{otherwise.} \end{cases}$$

[1] Most practical applications do not consider individual flow costs but they can be useful to force strong convexity with respect to all flow variables.

Then the arc-path formulation of the multicommodity flow problem is

$$\min f(x) = \sum_{j=1}^{n} f_j(x_{0j}) + \sum_{k=1}^{K} \sum_{j=1}^{n} f_{kj}\left(\sum_{p=1}^{N_k} \pi_{kp}(j) x_{kp} \right)$$

$$\text{s.t.} \sum_{k=1}^{K} \sum_{p=1}^{N_k} \pi_{kp}(j) x_{kp} = x_{0j}, \quad \forall j \in E, \tag{6}$$

$$\sum_{p=1}^{N_k} x_{kp} = r_k, \quad \forall k, \tag{7}$$

$$0 \leq x_{0j} \leq c_j, \quad \forall j \in E, \tag{8}$$

$$0 \leq x_{kp}, \quad \forall k, \forall p, \tag{9}$$

where x_{kp} is the flow of commodity k through the pth path.

The simple network constraints are now expressed by (7) and we have the same capacity constraints (8) and coupling constraints (6). This formulation assumes an exhaustive enumeration of all paths for each pair (s_k, t_k). This is unrealistic since the number of paths grows exponentially with the problem dimensions. In practice, methods dealing with this formulation do not require such an explicit enumeration but rather, they include some iterative path generation procedure.

The reader interested in the effects of the formulation of the multicommodity network flow problems in the framework of decomposition, is referred to Jones et al. (1993).

3. Literature Overview

In this section, an attempt is made to give a synthetic overview of solution techniques in the literature for nonlinear convex multicommodity flow problems. To this aim we take into account some criteria which, in our point of view, characterize the approaches. These criteria are: the decomposition strategy, the multicommodity flow model, the technique used to solve master and subproblems, and the potential applications.

As is shown below, most approaches use a decomposition strategy. In consequence, a lot of them are based on duality and relaxation of the coupling constraints. The efficiency of dual schemes depends highly on the smoothness of the dual function. This is obtained with a strictly convex objective function of the problem. For instance, Nagamochi (1988) has studied the model where all the functions f_j and f_{kj} are strictly convex. Strictly convexifying the objective function is also possible as shown by Stern (1977) who solved the message routing problem considering the objective function

$$f(x) = \sum_{j=1}^{n} \frac{x_{0j}}{c_j - x_{0j}} + r \sum_{k=1}^{K} \sum_{j=1}^{n} x_{kj}^2, \tag{10}$$

where r should be small enough to keep the solution as realistic as possible. Because of the importance of the characteristic of the objective function, we shall distinguish the following model types according to the objective function:

1. f convex;
2. f convex and differentiable,
3. $f(x) = \sum_{j=1}^{n} \frac{x_{0j}}{c_j - x_{0j}}$ or $f(x) = \sum_{j=1}^{n} f_j(x_{0j})$, f_j strictly convex, $f_{kj} \equiv 0$;
4. $f(x) = \sum_{j=1}^{n} f_j(x_{0j}) + \sum_{k=1}^{K} \sum_{j=1}^{n} f_{kj}(x_{kj})$, f_j, f_{kj} strictly convex.

The second column of Table 1 refers to these labels. It also indicates which formulation—node-arc or arc-path—is used in the proposed algorithms. As already noted, the multicommodity flow problem consists in determining a minimal nonlinear convex cost multicommodity flow through a network that meets the demand for each commodity subject to arc capacities restrictions, and flow conservation at transshipment nodes of the network. Other constraints are sometimes added to the models of § 2. One common example is to restrict individual commodity flows on the arcs (see for example Nagamochi 1988).

The multicommodity flow problems of § 2 may be solved by mathematical programming tools. Unfortunately, even for graphs of moderate sizes, the problem to be solved has exceedingly large dimensions. For real-life problems, the models may have tens or hundreds of thousands of constraints and hundreds of thousands or millions of variables: Direct approaches seem to lie beyond the capabilities of all existing softwares for convex optimization.

In contrast, the methods we review in this paper take advantage of the structure of the problem in one way or another; many of them are based on decomposition. The main motivation for decomposition is to reduce the problem to smaller subproblems, but other important motivations are present in multicommodity flow problems, namely to identify easier submodels as linear or convex minimum cost flow problems or shortest paths calculations, to parallelize or distribute computations among commodities, arcs or paths. The latter motivation has renewed interest for decomposition methods since the eighties with the increased development of parallel and distributed architectures (see Schultz and Meyer 1991). Approaches that have been effectively coded in parallel architectures are indicated in Table 1 by the symbol "//". The decomposition strategy relies essentially on a problem manipulation: dualization (see for example Goffin et al. 1997, Fukushima 1984), distributed coupling (Chifflet et al. 1994), and monotropic network programming (Mahey et al. 1998).

Concerning the techniques, early approaches were based on classical mathematical programming algorithms that were adapted to the convex multicommodity flow problem (steepest descent, Newton methods, conjugate gradient methods). In particular, the most popular among existing algorithms such as Flow Deviation (Fratta et al. 1973, LeBlanc 1973) and Projected Newton (Bertsekas and Gafni 1973) fall into the category of feasible direction methods. Linear or piecewise linear approximation of the nonlinear function (see Kennington 1978) and modification of the objective function (LeBlanc 1973, Bertsekas and Gallager 1987) were also proposed. On the other hand, some attempts have been made to solve the problem from a dual or primal-dual point of view. It is the case of methods that use proximal techniques (Chifflet et al. 1994, Eckstein and Fukushima 1993, Mahey et al. 1998), dual relaxation (Authie 1987, Stern 1977, Nagamochi 1988), cutting plane (Goffin et al. 1997), and subgradient methods (Fukushima 1984).

For the underlying applications, we have only retained those that have been originally treated in the

Table 1 A Summary of Some Approaches for Convex Multicommodity Flow Problems

Authors and References	Models Type and Formulations	Solution Techniques	Class	Subproblems	Decomposition	Applications
Fratta et al. (1973) LeBlanc (1973)	3 node-arc or arc-path	gradient method	primal	shortest paths problems	by commodity	telecom.
Cantor et al. (1974)	3 node-arc	simplicial decomp.	primal	shortest paths problems	no	telecom.
Schwartz et al. (1976)	3 node-arc	gradient projection	primal	descent direction	no	telecom.
Stern (1977)	4 node-arc	dual relaxation	dual	update formulae	by node	telecom.
LeBlanc et al. (1975)	3 node-arc	gradient, PARTAN's tech.	primal	shortest paths problems	by commodity	network equilibrium
Bertsekas et al. (1984), Bertsekas and Gafni (1983)	3 arc-path	projection methods	primal	shortest paths problems	by commodity	telecom.
Fukushima (1984)	3 arc-path	nondiff. optimization	dual	shortest paths problems	no	transport
Authié (1987)	3 node-arc	relaxation	primal-dual	nonlinear min. cost flow	by commodity //	telecom.
Nagamochi (1988)	4 node-arc	dual relaxation	dual	descent direction	by node	
Schultz et al. (1991)	2 node-arc	interior points	primal	min. cost flow	by commodity //	distribution
Pinar et al. (1992)	2 node-arc	penalty, simplicial decomp.	primal	min. cost flow	by commodities //	distribution
Eckstein et al. (1993)	1 node-arc	proximal	primal-dual	one-dim. minimization & nonlin. min. cost flow	distributed on arcs & simple commodities //	transport
Chifflet et al. (1994)	3 node-arc	proximal	primal-dual	one-dim. minimization & quadrat. min. cost flow	distributed on arcs & simple commodities	telecom.
Goffin et al. (1997)	1 node-arc	cutting plane, interior point	dual	one-dim. minimization, & shortest paths prob.	distributed on arcs & simple commodities	
Mahey et al. (1998)	3 arc-path	proximal	primal-dual	one-dim. minimization, & shortest paths prob.	distributed on arcs & simple commodities	telecom.

papers. When no reference is given, it means that the tested networks have been randomly generated. Other references on generalized transportation problems are not quoted here (see Patriksson 1994).

4. Main Solution Techniques

As already mentioned, the solution techniques for nonlinear multicommodity flow problems were initially special cases of classical methods for general nonlinear optimization programming. Recent techniques such as proximal and interior point ones have appeared later. In this section, we describe a wide selection of existing algorithms with emphasis on four algorithms which are numerically tested in § 5. We limit the algorithmic description to these four methods and sketch the main characteristics of the other references with no attempt to get into the details. This is mainly to alleviate the technical part of the survey and we give references where the interested reader can find further details and convergence results. We do not aim at ranking all existing methods. Instead, we intend to illustrate the difficulties of solving large multicommodity flow models. Therefore, we put forth four particular methods. We performed computational tests with these algorithms alone and show the impact on the various performance measurements. As explained in § 5, the codes have been all written in C with the same data structure adapted to specific networks issued from telecommunication applications. The common model is the message routing problem (f_j given by (5) and $f_{kj} \equiv 0$).

4.1. The Flow Deviation Method (FD)

The Flow Deviation method is a primal method that has been simultaneously proposed for the message routing problem by LeBlanc (1973) and Fratta et al. (1973). It is a special case of the so-called Franck-Wolfe method (Franck and Wolfe 1956 and Zangwill 1969) for solving nonlinear optimization problems with linear constraints. To deal with capacity constraints, the delay function f_j is replaced by

$$\bar{f}_j(x_{0j}) = \begin{cases} f_j(x_{0j}) = x_{0j}/(c_j - x_{0j}) \\ \quad \text{if } x_{0j} \in [0, \rho c_j], \\ \psi(x_{0j})\,(\text{quadratic or linear}) \\ \quad \text{if } x_{0j} > \rho c_j. \end{cases} \tag{11}$$

The quadratic (or linear) function is chosen such that the values of f_j and ψ and their first two derivatives (first derivative if ψ is linear) coincide at at the point $x_{0j} = \rho c_j$ (in general, $\rho = 0.99$). Let $\bar{f}(x_0) = \sum_{j=1}^{n} \bar{f}_j(x_{0j})$.

The message routing problem can then be written as min $\bar{f}(x_0)$ s.t. (1) and (3). The method successively solves linearized subproblems to get a feasible descent direction. If x_0^t is a feasible total flow vector at the start of iteration $t + 1$, the subproblem is minimize $\nabla\bar{f}(x_0^t)^T(\sum_{k=1}^{K} y_k)$ subject to network and nonnegativity constraints. Letting y_0^t denote the optimal total flow, the direction $y_0^t - x_0^t$ is searched for an improved solution x_0^{t+1}.

With the special structure of the Constraints (2), the direction-finding subproblem is solved by computing shortest paths between each pair (s_k, t_k) and loading the required amount of flow onto the corresponding path. The link costs used when finding cheapest routes are the partial derivatives of the objective function evaluated at the current solution.

More specifically, the entire algorithm consists of the four following steps:

1. (Initialization) Find a feasible total flow x_0^0, choose a tolerance parameter $\epsilon > 0$ and set $LB := 0$, $t := 0$.

2. (Subproblem) For each commodity k, find a minimum first derivative length path between s_k and t_k (the first derivatives are evaluated at the current solution x_0^t) and let y_k^t be the flow vector obtained by loading the corresponding amount r_k onto this path. Then, update the total vector flow $y_0^t := \sum_k y_k^t$.
(Step length search) Determine $\alpha_t := \arg\min \{\bar{f}(x_0^t + \alpha(y_0^t - x_0^t)),\ \alpha \in [0, 1]\}$.

3. (Flow deviation) $x_0^{t+1} := (1 - \alpha_t)x_0^t + \alpha_t y_0^t$.

4. (Termination criterion) Compute the lower bound[2]

$$LB := \max(LB, \bar{f}(x_0^t) + \nabla\bar{f}(x_0^t)^T(y_0^t - x_0^t)).$$

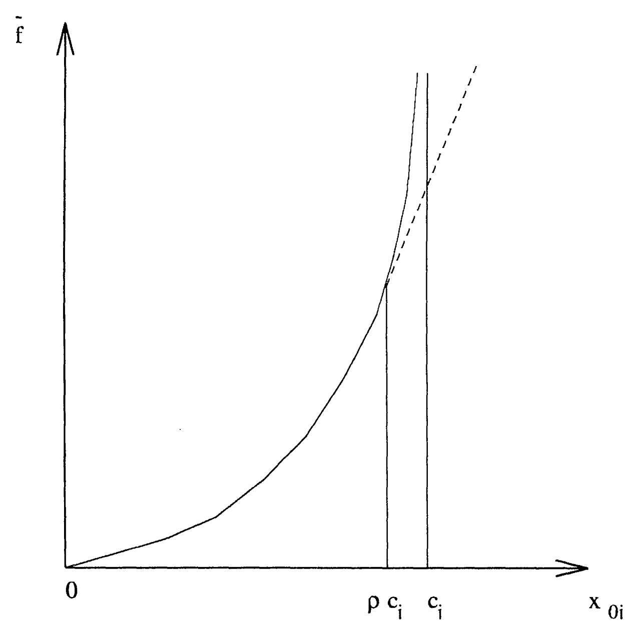
Stop the procedure if $\bar{f}(x_0^{t+1}) \le (1 + \epsilon)LB$; otherwise set $t := t + 1$ and go to Step 2.

To retrieve the complete routes taken by each commodity, one should simply store the the individual paths generated during the iterations. Should the optimal total link flows be the only quantities of interest, then the implementation of the flow deviation method requires a small amount of storage and thus allows to solve very large network problems. The characteristic property of the method is that flow is shifted from the nonshortest paths in equal proportions. This property distinguishes the flow deviation method from the method discussed in the next section. Some variants of the method have been proposed in the literature (LeBlanc et al. 1985, Florian et al. 1987). We also mention that an algorithm for finding a feasible starting multicommodity flow can be found in Kleinrock (1972).

Schultz and Meyer (1991) deal with a block angular optimization problem (which, of course, includes the convex multicommodity flow problem of § 2). They use a logarithmic barrier function to treat the coupling constraints. Then the barrier problem is solved by the method of Frank-Wolfe, which, as we have already noted, takes advantage of the constraints structure.

[2] If x^* denotes an optimal solution, from the facts that $\bar{f}$ is convex and that y^t solves the linearly approximated problem, we have, respectively, $\bar{f}(x_0^*) \ge \bar{f}(x_0^t) + \nabla\bar{f}(x_0^t)^T(x_0^* - x_0^t)$, and $\bar{f}(x_0^*) \ge \bar{f}(x_0^t) + \nabla\bar{f}(x_0^t)^T(y_0^t - x_0^t)$ (which is a lower bound of the optimal value).

Figure 1 Modified Objective Function in Flow Deviation Method



The search direction is a bit different here: The block structure of the problem allows a multidimensional search, that is a K-dimensional optimization problem with simple bounds is solved to coordinate the subproblem solutions. Some computational results are reported on test problems referred to as the Patient Distribution Systems (PDS) problems in the literature.

4.2. Projection Method (PM)

The flow deviation method tends to keep all generated path flows strictly positive. Such a behavior is improved in the so-called projection methods (Bertsekas and Gallager 1987, Bertsekas and Gafni 1983, Bersekas et al. 1984) in which the flow decreases along a nonshortest derivative path proportionally to the difference between its length and that of the shortest path. If such a decrease results in a negative flow, the path flow is simply set to zero.

Let $x_k^t = \{x_{kp}^t\}$ be the vector of path flows for commodity k at an iteration t and x_0^t the corresponding total vector flow. For each commodity, at that iteration, we have a set of active paths consisting of at most $(t - 1)$ paths. These paths were generated in the previous iterations and all nonactive paths carry zero flow. For the purpose of the next iteration $t + 1$, the message routing problem is converted to a problem involving only positive constraints. The capacity constraints are treated as in the flow deviation method (see (11)). Let path$_{k\bar{p}}$ be a shortest path between s_k and t_k with link costs $\bar{f}_j'(x_{0j}^t)$. From the network constraints $\sum_{p=1}^{N_k} x_{kp} = r_k$, we have $x_{k\bar{p}} = r_k - \sum_{p\neq\bar{p}} x_{kp}$; thus, direct substitution into the objective function yields the equivalent problem

$$\min\{\tilde{f}(\tilde{x}) \,|\, \tilde{x}_{kp} \geq 0 \text{ for all } k,\ p \neq \bar{p}\}, \tag{12}$$

where $\tilde{x}$ is the vector of all path flows which are not shortest paths ($\tilde{x}_{kp} = x_{kp}$ for all k, $p \neq \bar{p}$), and

$$\tilde{f}(\tilde{x}) = \sum_j \bar{f}_j\left(\sum_k \sum_{p:j\in\text{path}_{kp}} \tilde{x}_{kp}\right).$$

A projected Newton step is then applied to (12) (See Bertsekas 1982 for more general results on these convex problems):

$$\tilde{x}_{kp}^{t+1} = \max\left\{0,\ \tilde{x}_{kp}^t - \alpha_t\left[\frac{\partial^2\tilde{f}(\tilde{x}_{kp}^t)}{(\partial\tilde{x}_{kp})^2}\right]^{-1}\frac{\partial\tilde{f}(\tilde{x}_{kp}^t)}{\partial\tilde{x}_{kp}}\right\}.$$

Expressions for both the first and second derivatives of $\tilde{f}$ are straightforward and the path flows for the iteration $t + 1$ are updated according to:

$$x_{kp}^{t+1} = \max\left\{0,\ x_{kp}^t - \frac{\alpha_t}{H_{kp}}(d_{kp} - d_{k\bar{p}})\right\} \text{ for all } k,\ p \neq \bar{p}, \tag{13}$$

where d_{kp} and $d_{k\bar{p}}$ are first derivative lengths of path$_{kp}$ and path$_{k\bar{p}}$ given respectively by:

$$d_{kp} = \sum_{j\in\text{path}_{kp}} \bar{f}_j'(x_{0j}^t),\quad d_{k\bar{p}} = \sum_{j\in\text{path}_{k\bar{p}}} \bar{f}_j'(x_{0j}^t),$$

and

$$H_{kp} = \sum_{j\in L_{kp}} \bar{f}_j''(x_{0j}^t)$$

with $L_{kp} = (\text{path}_{kp} \cup \text{path}_{k\bar{p}})\backslash(\text{path}_{kp} \cap \text{path}_{k\bar{p}})$.

The path flows of the shortest paths are then adjusted so that for each commodity k, the sum of flows of all active paths equals the requirement r_k:

$$x_{k\bar{p}}^{t+1} = r_k - \sum_{p\neq\bar{p}} x_{kp}^{t+1} \quad \text{for all } k. \tag{14}$$

To summarize, after an initialization procedure (which consists of finding an initial feasible solution, as in the flow deviation method), the following steps are executed sequentially:

1. Compute a shortest path joining (s_k, t_k) for each commodity k with length $f'_j(x^t_{0j})$ on link j where x^t is the current solution. The shortest paths are added to the corresponding active paths for each k, if they are not already in the list.

2. The path flows are updated using (13). The shortest path flows are then adjusted according to (14). The stepsize α_t may be chosen by a variety of methods (see Bertsekas and Gallager 1987 and Bertsekas et al. 1984). Note that all the nonshortest path flows that are zero will stay at zero. Hence, the computations of the path flows are to be made for paths that carry positive flow.

Another projection method has been proposed by Schwartz and Cheung (1976) for the node-arc formulation of the message routing problem. The projection operator in this approach incorporates the constraint equations. As the authors pointed out, this algorithm is well suited to networks with a small number of commodities. Computational experiments indicate that for a large number of commodities, the flow deviation method performs better than this projection method.

4.3. Cutting Plane Algorithms

When formulated as a mathematical program, the nonlinear multicommodity flow problem has a constraint matrix with a primal block-angular structure. Decomposition algorithms strive to exploit this structure. First, using Lagrangian duality, the problem can be transformed into a nondifferentiable problem of much smaller size. Next, some specialized algorithm for nondifferentiable optimization is used to solve the transformed problem.

4.3.1. Decomposition Principle. Consider the node-arc formulation of the message routing problem. Since the objective function is strictly increasing in x_{0j}, the coupling equality Constraint (1) can be replaced by $\sum_{k=1}^{K} x_{kj} \leq x_{0j}$. Associating with the coupling Constraints (1) the dual variables $u \geq 0$, we construct the partial Lagrangian:

$$\mathcal{L}(x; u) = \sum_{j=1}^{n} f_j(x_{0j}) + \sum_{j=1}^{n} u_j \left(\sum_{k=1}^{K} x_{kj} - x_{0j} \right). \quad (15)$$

We associate the dual problem

$$\max_{u\geq 0} L(u) = \max_{u \geq 0} \left(\sum_{j=1}^{n} L_{0j}(u) + \sum_{k=1}^{K} L_{1k}(u) \right), \quad (16)$$

where

$$L_{0j}(u) = \min\{f_j(x_{0j}) - u_j x_{0j} | 0 \leq x_{0j} \leq c_j\}, \qquad j = 1, \ldots, n \quad (17)$$

are simple n nonlinear one-dimensional subproblems, and

$$L_{1k}(u) = \min_{x_k \geq 0} \left\{ \sum_j u_j x_{kj} : Mx_k = r_k b_k \right\}, \quad k = 1, \ldots, K \quad (18)$$

are K shortest paths problems with arc costs u_j. Both types of problems are easily solved. In particular, the solution of Problem (17) is given by the equation $f'_j(x_{0j}) = u_j$. A closed form solution for the delay function (5) can be readily computed. From duality theory, the optimal value of Problem (16) is equal to the optimal solution of the primal problem.

As a by-product of the computation of the L_{0j} and L_{1k} one easily obtains an element of the subdifferential. Indeed, let x_{kj} be an optimal solution for the value u. Then, for any $v \geq 0$ one has $L_{1k}(v) \leq \sum_j u_j x_{kj} = L_{1k}(u) + \sum_j x_{kj}(v_j - u_j)$. Hence, x_k is in the subdifferential set. A similar reasoning applies to L_{0j}. The subdifferential elements can be concatenated in a single vector $\xi \in -\partial(-L(u))$ and the inequalities are summarized into the valid inequality

$$L(v) \leq L(u) + \xi^T(v - u). \quad (19)$$

In conclusion, we transformed the initial problem into a nondifferentiable convex problem, $\max_u L(u)$, of much smaller size. Besides, the valid inequality (19) can be used to generate a polyhedral outer approximation of L.

4.3.2. A Generic Cutting Plane Method. Many different strategies exist to solve the nondifferentiable

Problem (16). Let us mention a few of them that have been applied in the context of multicommodity flow problems: The bundle method (Lemaréchal 1974, Wolfe 1975) used by Medhi (1994), the standard cutting plane method (Cheney and Goldstein 1959, Kelley 1960, Dantzig and Wolfe 1961) used by e.g., Jones et al. (1993), and the analytic center cutting plane method (Goffin et al. 1992), thereafter named ACCPM. We shall start with a generic description of cutting plane methods, and then focus on ACCPM.

Cutting plane methods are based on increasingly refined polyhedral approximations of the epigraph of $-L(u)$.[3] We give a very general description of those methods. Let $\{u^t\}_{t=1}^T$ be a sequence of query points. At each point, one computes $L(u^t)$ and a subgradient $\xi^t \in \partial L(u^t)$. As mentioned in the previous subsection, the subgradients are direct by-products of the optimization in (17) and the shortest path computations. Consequently, the program

$$\max\{z | z \le L(u^t) + (\xi^t)^T(u - u^t),\ t = 1,T\} \tag{20}$$

is a polyhedral relaxation of (16). It is worthwhile noticing that any feasible solution for the dual of (20) provides an upper bound for $\max_u L(u)$. Also,

$$\theta_T = \max_{t=1,\cdots T} \{L(u^t)\}$$

is a valid lower bound for the optimal value. Therefore, at any stage the optimal value can be bracketed. One can associate with the lower bound θ_T the so-called localization set

$$\begin{aligned} H_T = \{(u, z) | z \ge \theta_T,& \\ z \le L(u^t) + (\xi^t)^T(u - u^t),&\ t = 1,T, \\ 0 \le u_j \le U_j,&\ j = 1,n\}. \end{aligned} \tag{21}$$

The upper bound constraint $u \le U$ is introduced to enforce compactness. U is supposed to be large enough to make the constraint inactive at the optimum. Note that the localization set always contains the optimal solution of the original problem.

[3] In our formulation $L(u)$ is concave. Its epigraph is not convex, but the epigraph of $-L(u)$ is convex.

The basic step of the generic cutting plane is as follows. Assume H_T is given.

1. Pick $u^{T+1} \in H_T$ and compute an upper bound via (20).
2. Compute $L(u^{T+1})$ and the valid inequality (19).
3. Update the localization set $H_{T+1} := H_T \cup \{z \le L(u^{T+1}) + (\xi^{(T+1)})^T(u - u^{T+1})\}$.
4. Update the upper and the lower bounds.
5. Check if the distance between the upper and lower bounds is below the stopping threshold value.

4.3.3. The Analytic Center Cutting Plane Method (ACCPM). The standard cutting plane method (Cheney and Goldstein 1959, Kelley 1960, Dantzig and Wolfe 1961) defines the next query point u^{T+1} as the maximizer of (20). Let us point out that the constraints of the dual of (20) can be interpreted as a convex combination of the matrix columns, i.e., of the shortest paths. One can therefore easily reconstruct a primal solution, which, however, may not satisfy the coupling constraint restriction.

The standard cutting plane is reputed unstable. Many regularization schemes have been proposed. Among them, one can find so-called central methods, which select in Step 1 of the generic cutting plane method some kind of center of the localization set. Not only does the choice regularize the algorithm, but often it allows to derive complexity estimates. Among possible centers, the analytic center is quite favorable because it is easily computed by interior point methods and because one can construct a pseudo-polynomial complexity estimate.

The analytic center is defined as the unique minimizer of the potential

$$-\log(z - \theta_T) - \sum_{t=1}^{T} \log(L(u^t) + (\xi^t)^T(u - u^t) - z)$$

$$- \sum_{j=1}^{n} (\log u_j + \log(U_j - u_j)).$$

Interior point methods apply to this type of problem. The iterative step is a damped Newton step. One obtains dual variables as a by-product of the computations. At (approximate) centers, those dual variables become feasible to the dual of (20). One can thus construct upper bounds for the optimal value of the original problem.

In practical implementations, it is best not to concatenate the subdifferentials of L_{0j} and L_{1k} but to introduce as many valid inequalities as there are functions L_{0j} and L_{1k}, i.e., the total number of arcs and commodities. This multicut approach is much more efficient (see du Merle et al. 1998). The ability of the method to compute new analytic centers after adding many cuts is an important implementation issue. It turns out that the method requires only a few Newton iterations to recompute an analytic center. However, each iteration remains computationally costly. For a detailed description of the method in the context of nonlinear multicommodity flow problems, we refer to Goffin et al. (1997).

Let us mention that ACCPM has been extensively tested on very large problems (Goffin et al. 1997, Gondzio et al. 1997).

4.4. The Proximal Decomposition Method (PDM)

Consider the arc-path formulation of the message routing problem. Let $u_0 \in \mathbb{R}^n$ and $u_1 \in \mathbb{R}^K$ be the dual vectors associated with the Constraints (6) and (7), respectively. The dual problem may then be written as

$$\max_{u_0 \in \mathbb{R}^n, u_1 \in \mathbb{R}^K} \mathcal{F}(u_0, u_1), \tag{22}$$

where

$$\mathcal{F}(u_0, u_1) = \min_{0 \le x_{0j} \le c_j, 0 \le x_{kp}} \left\{ \sum_{j=1}^{n} \left(f_j(x_{0j}) - u_{0j}\left(x_{0j} - \sum_{k=1}^{K} \sum_{p=1}^{N_k} \pi_{kp}(j) x_{kp} \right)\right) + \sum_{k=1}^{K} u_{1k}\left(r_k - \sum_{p=1}^{N_k} x_{kp} \right) \right\}.$$

To apply efficiently the proximal decomposition algorithm (Mahey et al. 1995), we make a copy of the dual vector u_0 for each path $p = 1, \ldots, N_k$ between origin-destination pair (s_k, t_k) for commodity k, and a copy of the dual variable u_{1k} for each path $p = 1, \ldots, N_k$. We denote these copies respectively by u_{0kp} and u_{1kp}; the components of u_{0kp} are denoted by $u_{0kp}(j)$. We then introduce the subspace

$$A = \{u = (u_0, u_{0kp}, u_{1kp}): u_0, u_{0kp} \in \mathbb{R}^n, u_{1kp} \in \mathbb{R}, u_0 = u_{0kp}, k = 1, K, p = 1, N_k, u_{1kp'} = u_{1kp}, k = 1, K, p \neq p'\},$$

to write the dual problem in the form

$$\max_{u \in A} F(u), \tag{23}$$

where the objective function is now separable with respect to u_0 and (u_{0kp}, u_{1kp}):

$$F(u) = \sum_{j=1}^{n} \min_{0 \le x_{0j} \le c_j} \{f_j(x_{0j}) - u_{0j}x_{0j}\} + \sum_{k=1}^{K} \sum_{p=1}^{N_k} \min_{0 \le x_{kp}} \left\{ \left(\sum_{j=1}^{n} \pi_{kp}(j) u_{0kp}(j) - \frac{r_k}{N_k} u_{1kp} \right) x_{kp} \right\}.$$

The proximal decomposition Algorithm is a specialized version of the partial inverse method designed by Spingarn (1983) for constrained programs of the form (23). If $A^\perp$ denotes the orthogonal subspace to A, an optimal primal-dual pair $(\bar{u}, \bar{v})$ must lie in the Cartesian product space $A \times A^\perp$. The algorithm performs two distinct steps at each iteration: a proximal step that regularizes the objective function by adding a quadratic term depending on the previous primal-dual pair of solutions, and a projection step on the corresponding subspaces. More precisely, given a primal-dual pair $(u^t, v^t) \in A \times A^\perp$ and a positive parameter λ, the computation of the new updates (u^{t+1}, v^{t+1}) is performed as follows:

proximal step:

$$y^t = \arg\max_y \left\{ F(y) + \frac{1}{2\lambda} \left\| y - u^t - \lambda v^t \right\|^2 \right\},$$

$$z^t = \lambda^{-1}(u^t + \lambda v^t - y^t).$$

projection step: $(u^{t+1}, v^{t+1}) = \mathrm{proj}_{A \times A^\perp}(y^t, z^t)$.

We skip the technical steps that lead to the distribution of the computations on each commodity, each arc, and each individual path, and updates of the dual variables (see Ouorou 1995 for the intermediate steps). As the set of paths between s_k and t_k is not known a priori, we substitute it at each iteration $t = 0, 1, \cdots$ by a subset which contains the previously generated paths. The proximal step consists of one-dimensional convex subproblems for each arc to find aggregate flows x_{0j}^{t+1}. Then, new paths are generated by shortest paths calculation with link costs $f_j'(x_{0j}^{t+1})$ followed by a distributed updating for path flows and potentials. The whole algorithm is presented below with the following notation: N_k^t denotes the number of paths corresponding to the commodity k at iteration t, $d(j)$ denotes the number of paths sharing j and the residual (violation of Constraints (6) and (7)) for a vector $x = (x_k, k = 0, \ldots, K)$ is denoted by:

$$r_j(x) = \sum_{k=1}^{K} \sum_{p=1}^{N_k} \pi_{kp}(j) x_{kp} - x_{0j}$$

and

$$r_k(x) = r_k - \sum_{p=1}^{N_k} x_{kp}.$$

1. Choose the convergence parameters $\epsilon_1, \epsilon_2, \lambda > 0$. Set the iteration index $t := 0$. The initial vectors x^0, u_0^0, u_1^0 may be chosen arbitrarily.

2. For each arc j compute

$$x_{0j}^{t+1} := \arg \min_{0 \le x_{0j} < c_j} \left\{ f_j(x_{0j}) - u_{0j}^t x_{0j} + \frac{\lambda}{2} \left((x_{0j})^2 - 2\left(x_{0j}^t + \frac{r_j(x^t)}{d(j)} \right) x_{0j} \right) \right\}.$$

3. For each commodity k, compute the shortest path that joins the origin s_k and the destination t_k. The length for each arc j used for this computation is $f_j'(x_{0j}^{t+1})$. This shortest path is added to P_k^t and $N_k^t := N_k^t + 1$ if it is not already there. Then, the path flows are updated according to the following rule:

$$x_{kp}^{t+1} := \max\left(0, x_{kp}^t + \frac{1}{\lambda(1 + \|\pi_{kp}\|^2)} \left(u_{1k}^t - \sum_{j \in pa_{kp}} u_{0j}^t \right) + \frac{1}{1 + \|\pi_{kp}\|^2} \left(\frac{r_k(x^t)}{N_k^t} - \sum_{j \in pa_{kp}} \frac{r_j(x^t)}{d(j)} \right) \right).$$

4. Update the dual variables

$$u_{0j}^{t+1} := u_{0j}^t + \frac{\lambda}{d(j)} r_j(x^{t+1}), \quad u_{1k}^{t+1} := u_{1k}^t + \frac{\lambda}{N_k^t} r_k(x^{t+1}).$$

5. Test $(x_{0j}^{t+1}, x_{kp}^{t+1}, u_{0j}^{t+1}, u_{1k}^{t+1})$ for convergence and set $t := t + 1$ if one decides to continue the iteration.

These five steps constitute the heart of the (PDM) algorithm which will be used in the tests below. Its good performance relies mainly on the arc-path formulation. The one-dimensional minimizations at Step 2 can be easily solved by the Newton method.

Classical node-arc formulation has been proposed earlier: in Chifflet et al. (1994), the subspace A represents the coupling between commodities (Equation (1)). Then, the proximal step splits into one-dimensional convex programs for each arc and minimum quadratic cost flow subproblems for each commodity. Hence, the flow conservation equations for all commodities are always satisfied. This feature turns out to be very useful in practical applications. The projection step consists of simple updates of the dual variables. Although this algorithm is efficient for solving these problems, it is very slow when the number of commodities is large. One reason why this happens is that the algorithm involves at each iteration quadratic flow subproblems. See Mahey et al. (1998) for a comparison with the above PDM algorithm.

Eckstein and Fukushima (1993) give some reformulations of the generalized alternating direction method of multipliers by Eckstein (1989). These reformulations are then applied to the multicommodity flow problem and other optimization problems. The resulting algorithm for the multicommodity flow problem is very close to the one derived from the proximal decomposition method in Chifflet et al. (1994) as it splits into one-dimensional convex programs for each arc and single-commodity minimum cost flow problems for each commodity. The method has been implemented with the data-parallel CM Fortran on the Connection

Machine 5, and applied to some randomly generated quadratic transportation problems.

Another interesting application of the proximal point algorithm is given by Ibaraki and Fukushima (1994). The primal-dual algorithm which is proposed for a large class of convex multicommodity flow problems, resembles in the separable case, to those proposed in Chifflet et al. (1994) and Eckstein and Fukushima (1993). At each iteration, the flow conservation equations are satisfied and the dual optimality is attained when the coupling constraints are satisfied. Computational experiments were performed on multicommodity flow test problems with separable quadratic costs.

4.5. Other Methods

Stern (1977) groups the variables by destination which allows us to have a linearly independent set of flow constraints for each destination. In this relaxation method, the primal variables must be uniquely defined as functions of the dual variables. To achieve that, it is sufficient to use a strictly convex objective function with respect to all primal variables. Consequently, the objective function considered is of the form (10). The proposed method is of a Gauss-Seidel type: Dual vectors corresponding to each node are computed sequentially while the others are kept fixed. Parallel asynchronous versions are also discussed in Bertsekas and Tsitsiklis (1989). An example is presented to illustrate the method.

Nagamochi has developed a relaxation method for the nonlinear multicommodity flow problem which requires the functions f_j and f_{kj} to be strictly convex. Hence, to apply this method to the message routing problem, its objective function must be modified as suggested by Stern (1977), for example (see (10)). In the model he is concerned with, capacity constraints are added to individual commodities. The proposed method is an extension of the relaxation method of Bertsekas for network flow problems with separable strictly convex costs (Tseng and Bertsekas 1987). Computational experiments have been conducted on randomly generated test problems where all the functions f_j and f_{kj} are quadratic.

The method proposed by Authie (1987) fits in the general relaxation framework. It consists of solving sequentially single commodity flow problems when the remaining commodity flows are kept fixed to their former values, like in the block-coordinate descent method (see Bertsekas and Tsitsiklis 1989). The single commodity flow problems are solved by a dual approach. Some computational experiments show that the method is comparable to the approach of Schwartz and Cheung (1976).

Fukushima's (1984) approach is an adaptation of a nondifferentiable optimization technique to the dual of the arc-path formulation of the multicommodity flow problem with f_j strictly convex and $f_k \equiv 0$. The dual problem is shown to be a problem of maximizing a concave function for which functional values and subgradients can be calculated by using shortest path algorithms. A nonsmooth optimization algorithm of descent type is proposed which takes special features of the dual problem into account. The algorithm has been tested on traffic assignment problems.

Pinar and Zenios (1992) considered a block angular optimization model with a smooth objective function (as Schultz and Meyer 1991). Their method uses a piecewise linear-quadratic penalty function to eliminate the coupling constraints. This penalty function is continuous, smooth, and convex but not separable. The special structure of the constraints motivates the use of a method based on linearization of the penalty function to solve the penalized problem. Simplicial decomposition (von Hohenbalken 1977) has been chosen by the authors. See Pinar and Zenios (1993) for a comparative study of this algorithm with two other algorithms for multicommodity flow problems.

The Minimum Mean Cycles Canceling method proposed by Ouorou and Mahey (1996), is a primal method which is inspired by Karzanov and McCormick's idea for separable convex optimization in unimodular linear spaces (Karzanov and McCormick 1993), first worked out by Goldberg and Tarjan (1989) for minimum cost circulations. The authors deal with the node-arc formulation where each f_j is continuously differentiable and $f_{kj} \equiv 0$. The method starts with a feasible multicommodity flow and at each iteration, a commodity flow with positive *absolute mean* (see Ouorou and Mahey 1996) is then chosen and a negative cycle, feasible with respect to that commodity, is found and canceled. The canceling step increases the total flow and the flow of the chosen commodity on forward arcs in the feasible cycle and decreases them on backward arcs. It maintains the feasibility of the new commodity and decreases the cost function. As the method needs to store all individual flow vectors, it is suited to networks with a limited number of commodities.

5. Numerical Experience

Computational experiments have been conducted for most of the techniques discussed in the previous section. But as we already noted, they were run with different codes and test problems. However, in this section, we attempt to give a performance evaluation and computational testing of some chosen solution techniques using an actual telecommunication network with 106 nodes and 904 arcs and dense requirement matrix[4]. (Recall that much of our motivation comes from the message routing problem.) We use different densities of the requirement matrix (from 40% to a fully dense matrix, i.e., 11,130 = 106 $\times$ 105 commodities share the network for the latter case) and introduce a load factor (up to 3 times the standard demand; the load factor is used to scale up the r_ks) to have different offered traffic for this network, leading to 20 test problems. All generated problems were feasible. We complete these tests problems with a set of randomly generated feasible problems to have a sufficient variety of network structures.

We have chosen to test four solution techniques, and two primal algorithms: the Flow Deviation (FD) (LeBlanc 1973), and the Projection Method (PM) (Bertsekas et al. 1984, Bertsekas and Gallager 1987), and two primal-dual algorithms: the Proximal Decomposition Method (PDM) proposed in Mahey et al. (1998), and the Analytic Center Cutting Plane Method (ACCPM) (Goffin et al. 1997). Since there are many different performance criteria (speed, accuracy, stability, robustness, path dispersion), we do not attempt to rank the methods. Our goal is to give valuable information to help the network designer select the method that best fits his/her individual need.

All the computational tests were performed on an IBM RISC/System 6000 machine and the codes are entirely written in C.[5] This programming language is able to work properly with specific data structures. Double precision was used for all calculations. The candidate paths were generated using Dijkstra's shortest path algorithm and the same initialization procedure is used for FD and PM.

Stopping Tolerances. To begin with, we point out that the performance evaluation heavily relies on the precision measure which is used to stop the algorithms. It is not adequate to require that all algorithms reach the same final accuracy. The first reason is that the stopping criterion is not unique and differs from one method to another with varying effects on the final accuracies in constraint violations, Kuhn-Tucker conditions, and objective function values. The second reason is that some algorithms cannot achieve a higher accuracy (especially on our large testbed network) because they are not specially tailored for that quality criterion but for a different one, like very fast early convergence (e.g., FD) or ability to support distributed computations (e.g., PDM). Hence, we have chosen to stop each algorithm on its own optimality test.

At each iteration of the Flow Deviation algorithm, a lower bound on the optimal value is available. One can terminate the procedure when the best obtained lower bound during iterations is sufficiently close to the objective function value at the new current solution: all the results below are obtained with an accuracy of 1%. One alternative is to utilize Wardrop's equilibrium conditions directly, measuring the difference in the first derivative lengths of the paths used within an origin-destination pair (Patriksson 1994).

PM is stopped when a normalized measure of deviation from the optimal solution falls below a given tolerance, as proposed by the authors (Bertsekas et al. 1984). Here we use a tolerance of 10^{-3}.

For PDM, we stop computing when the maximal tolerance for both row residual and the other Kuhn-

[4] This material is issued from a real-world traffic situation which has been given by the Centre National d'Etudes des Télécommunications.

[5] The codes for the Flow Deviation and the Projection Method were originally written in FORTRAN.

Tucker optimality are respectively lower than 10^{-4} and 10^{-3} (see Mahey et al. (1998) for more details).

The case of ACCPM deserves a few lines of discussion. It does not directly solve the primal problem. Rather, it solves its Lagrangian dual (16), and provides a lower and upper bound values for the optimum of this problem. However, the practical solution involves additional box constraints on u. Any lower bound for this modified problem is a valid lower bound for the original one. One cannot assert that the same holds for the computed upper bound. But the method offers an alternative for computing a valid upper bound. The primal variables in the computation of the analytic center define feasible convex combinations of the path-flows for each commodity. We can thus construct a feasible flow, and the associated delay is a valid upper bound. This method of computing valid upper bounds for (16) requires a slight modification of ACCPM. Indeed, ACCPM is a general purpose code aimed at solving nondifferentiable problems such as (16). For our experiments, we have used the lower and the upper bounds generated by ACCPM, with the risk of getting invalid upper bounds. This was also the way things were done in Goffin et al. (1997). To check the risk of error, we run our main example (the actual network) with computation of the upper bound based on a feasible flow solution. We found that the delay was each time inferior to the proposed upper bound. The tolerance parameter for ACCPM is set to 10^{-5}.

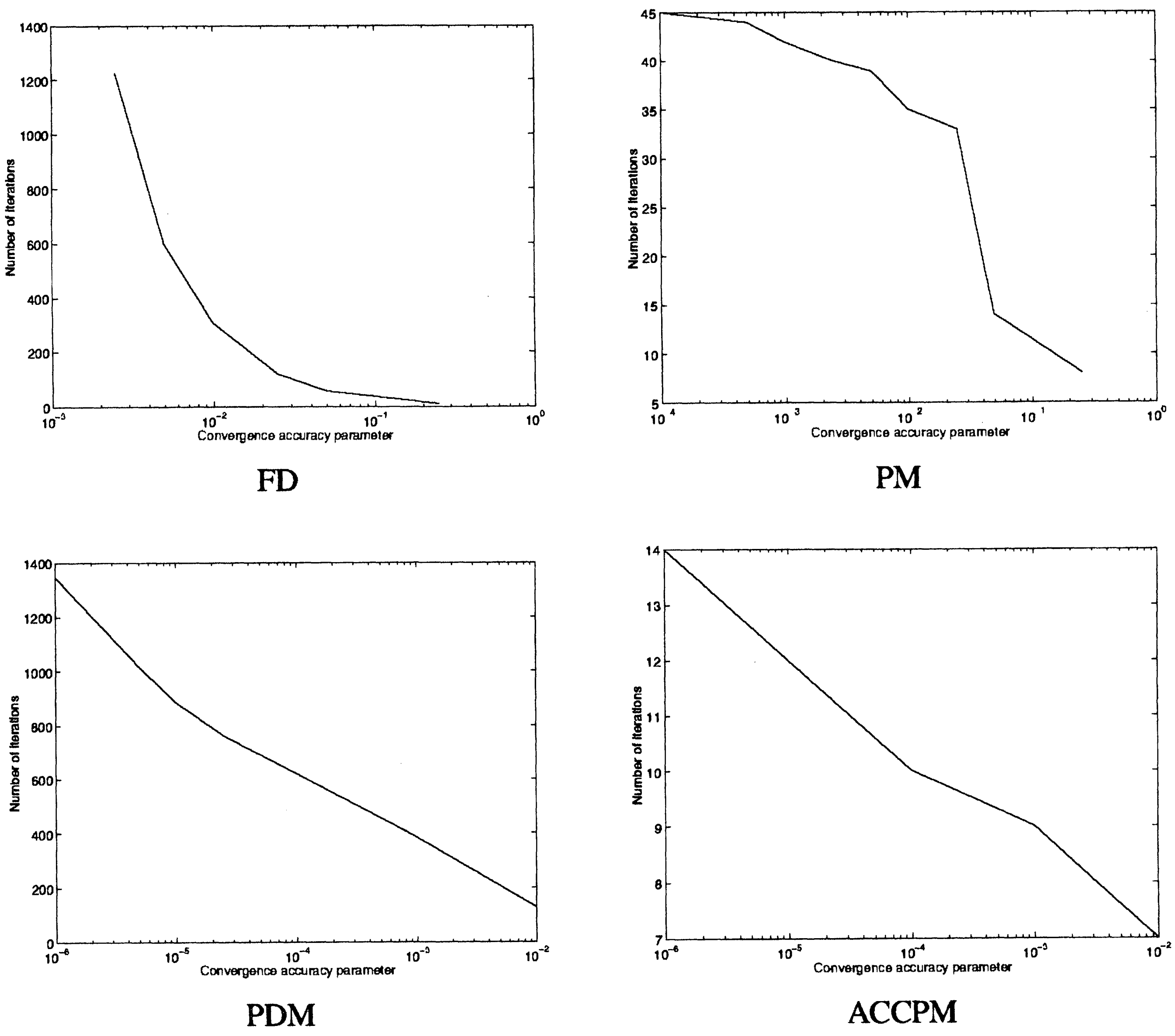
All these tolerances were chosen after several experiments which have shown that with these choices, the aggregate flow differs only slightly from one algorithm to another. Figure 2 displays a semilog plot of the number of iterations and the CPU time versus the accuracy tolerance. An actual 19 nodes-68 arcs network with 30 commodities has been used for these tests as high accuracy requires much time on large networks. Note also that with large networks it is not always possible (for reasons pointed out earlier) to require the algorithm to achieve some of the tolerances we use for the small network in a reasonable amount of computing time.

The Flow Deviation method has the reputation of having a slow convergence tail when a high accuracy is required. Its convergence is quick at the beginning and as we get closer to the optimal solution, the algorithm becomes slower. The PM algorithm progress is shown to be often satisfactory near a solution and better than that of FD (see Bertsekas and Gallager 1987, p. 472 for an illustration).

Path Dispersion. In the message routing problem, the paths flows are not unique although the total link flows are unique. Solutions with a large number of paths are less desirable than solutions which use only a few paths. On this basis, PDM is most likely to find solutions with a reduced number of paths, while the other algorithms always find a solution with a larger number of paths. The reason why PDM uses a fewer number of paths, can be explained intuitively by the update of the path flows procedure which tends to force many of them to zero. This intuition is here empirically confirmed by our experimentation (more details on path dispersion and performance of PDM can be found in Mahey et al. (1998). At each iteration of the FD method, a vertex of the feasible set is obtained by shortest paths computation (see § 4.1). Then the next iterate is obtained by a search along the line joining the current iterate with the vertex. The line search will keep all previously generated path flows strictly positive, unless the stepsize equals one which is unlikely in practical situations. This point explains why the Flow Deviation method tends to generate a large number of active paths. Observe that the behavior of FD to keep all generated path flow positive is improved by PM: An increment of flow change is calculated for each path on the basis of the relative magnitudes of the path lengths and the second derivatives of the cost function. If the increment is too large so that the path flow becomes negative, the path flow is set to zero. All these characteristics are observed in Figure 3 (for $K = 11{,}130$). In Tables 2 and 4, the column headed "Path Dispersion" gives the maximum number of paths used by a commodity.

Other Characteristics. The sensitivity of PDM w.r.t. the choice of the proximal parameter λ is illustrated in Figure 4. We have used the same value of the parameter $\lambda = 1$ in all the tests, but one can see that in Figure 4 the value could be good or bad depending on the data.

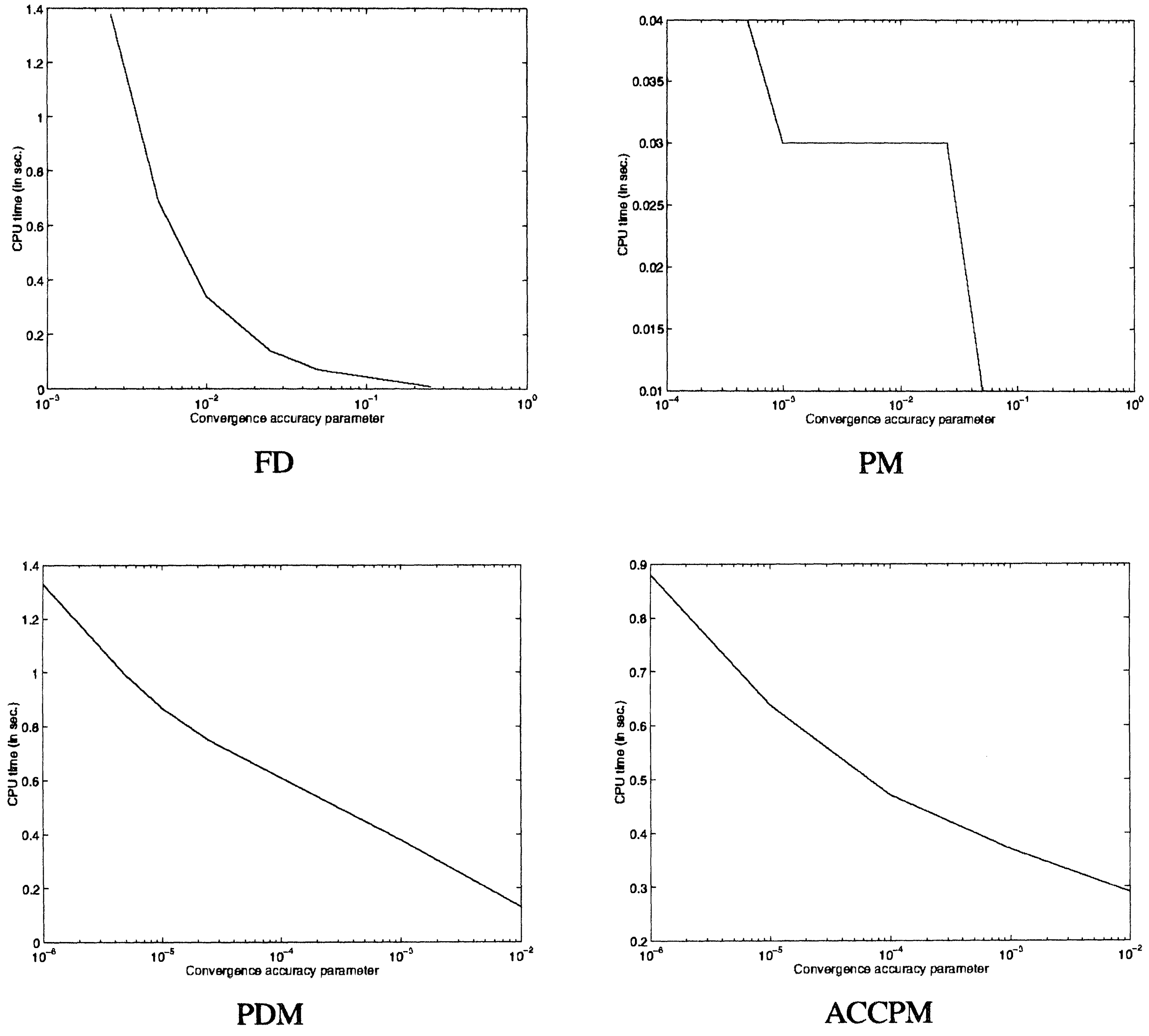
Figure 2 Number of Iterations and CPU Time w.r.t. Tolerance, Part 1



We would like to point out that at each iteration of FD and PM, the current solution is feasible, while for PDM, feasibility is not maintained during the iterations like for all dual methods. To implement ACCPM, we introduce the box constraint $0 \leq u_j \leq U$. At the optimum the upper constraint should not be active. In this paper we chose the default value $U = 10$. This was convenient since for most variables the optimal value was close to 0. In case $U = 10$ appears to be an underestimate for some variable, the box constraint is automatically expanded as in a trust region scheme.

Discussion. Tables 2 and 3 provide a detailed account of the results obtained with each algorithm on the 20 test problems (recall that K denotes the number of commodities) on the 106 nodes-904 arcs network. The times reported are in seconds and the column headed "Delay" corresponds to the values of (5)

Figure 2 Number of Iterations and CPU Time w.r.t. Tolerance, Part 2



computed for the solutions given by the algorithms. Since FD and PM are primal-feasible, "Delay" means the best recorded primal objective value. For PDM, it is the primal objective value of the final sum of the path flows. For ACCPM, it is computed using the reconstructed feasible solution as indicated above. Tables 2 and 3 display a great diversity in the methods even if PM and PDM seem to converge faster. ACCPM is significantly slower on these examples. These results also show how computational times increase with two characteristic factors: the traffic load and the number of commodities. CPU time is certainly the most popular criterion to evaluate an algorithm, but it must be put in perspective when observing that it is often nonmonotonic with respect to some parameters and data. For example, ACCPM looks quite insensi-

Figure 3 Path Dispersion Versus Traffic Load

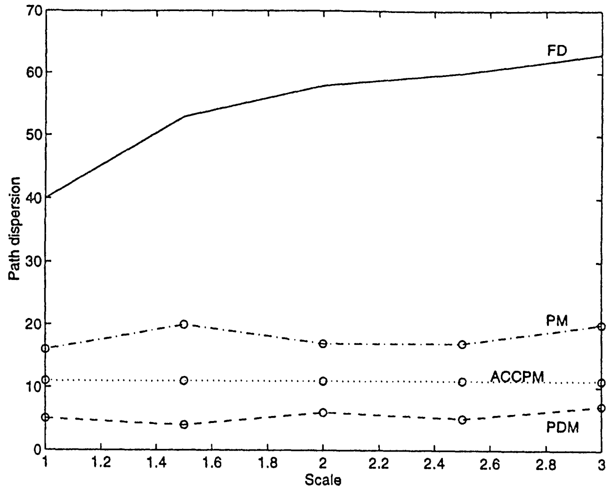
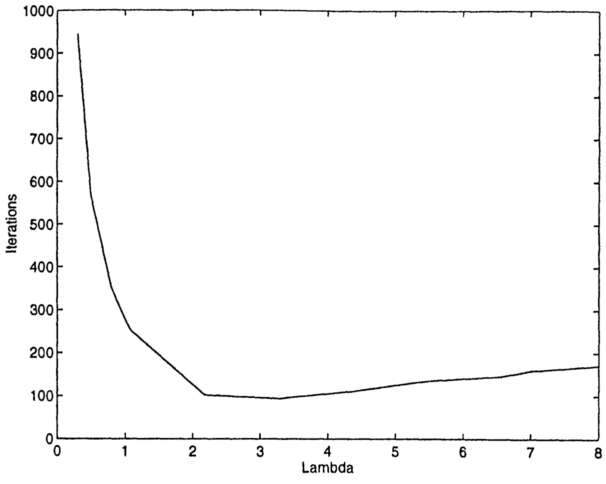


Figure 4 Sensitivity of PDM w.r.t. λ



tive to the problem size and to the load factor while the other three methods take more time to solve heavy congested networks. On the other hand, FD showed high sensitivity with respect to K while the other three methods seemed to be more robust when dealing with dense requirement matrices. It is worth noting that many authors have tested their algorithms on networks of increasing sizes, but very few used a full set of commodities as we do here.

Table 4 collects a representative subset of results obtained with the four algorithms on some other

Table 2 Summary of Results on the Test Problems Using the 106 Nodes-904 Arcs Network

Load Factor		Delay	CPU Time	Number of Iter.	Path Dispersion
		K = 4,452			
1.0	FD	12.5850	109.61	380	16
	PM	12.5846	47.82	185	9
	PDM	12.5894	51.56	209	3
	ACCPM	12.5846	2,185.04	15	11
1.5	FD	19.1808	188.25	643	17
	PM	19.1799	52.25	188	11
	PDM	19.1904	51.28	208	3
	ACCPM	19.1798	1,997.56	14	11
2.0	FD	25.9768	269.24	887	22
	PM	25.9755	101.26	345	12
	PDM	25.9934	62.01	253	4
	ACCPM	25.9754	1,918.72	14	11
2.5	FD	32.9826	353.29	1,180	22
	PM	32.9809	98.21	342	11
	PDM	33.0108	85.9	353	4
	ACCPM	32.9808	1,854.81	14	12
3.0	FD	40.2093	462.82	1,507	29
	PM	40.2072	143.79	460	13
	PDM	40.2526	96.45	392	3
	ACCPM	40.2072	2,262.37	14	11
		K = 6,678			
1.0	FD	19.6487	241.34	629	20
	PM	19.6481	115.7	312	10
	PDM	19.6546	90.85	265	3
	ACCPM	19.6481	2,251.27	11	10
1.5	FD	30.1790	419.53	1,066	26
	PM	30.1775	209.83	519	13
	PDM	30.1904	91.87	267	3
	ACCPM	30.1775	3,140.37	14	10
2.0	FD	41.2086	619.01	1,534	36
	PM	41.2065	197.18	478	12
	PDM	41.2315	175.64	519	3
	ACCPM	41.2065	3,577.17	14	10
2.5	FD	52.7817	828.18	2,090	36
	PM	52.7789	217.82	505	13
	PDM	52.8240	231.15	679	3
	ACCPM	52.7789	2,291.84	12	10
3.0	FD	64.9493	1,092.53	2,713	37
	PM	64.9461	164.28	390	12
	PDM	65.0173	390.14	1,024	4
	ACCPM	64.9460	3,396.46	14	11

Table 3 Summary of Results on the Test Problems Using the 106 Nodes-904 Arcs Network

Load Factor		Delay	CPU Time	Number of Iter.	Path Dispersion
		K = 8,904			
1.0	FD	26.4738	442.04	912	31
	PM	26.4730	259.33	518	13
	PDM	26.4792	505.34	960	5
	ACCPM	26.4730	2,973.46	14	12
1.5	FD	40.9763	737.39	1,503	47
	PM	40.9741	357.32	651	15
	PDM	40.9888	391.73	894	4
	ACCPM	40.9742	3,274.3	14	10
2.0	FD	56.4262	1,105.23	2,229	47
	PM	56.4232	340.65	609	18
	PDM	56.4569	326.27	738	4
	ACCPM	56.4232	3,190.93	14	10
2.5	FD	72.9429	1,597.0	3,214	51
	PM	72.9391	418.53	748	17
	PDM	73.0019	513.56	1,132	5
	ACCPM	72.9392	3,023.41	14	11
3.0	FD	90.6668	2,131.28	4,244	51
	PM	90.6620	397.88	699	14
	PDM	90.7651	617.19	1,381	5
	ACCPM	90.6620	3,105.18	14	11
		K = 11,130			
1.0	FD	33.4941	642.33	1,120	40
	PM	33.4930	379.9	579	16
	PDM	33.5006	456.19	845	5
	ACCPM	33.4930	3,233.59	14	11
1.5	FD	52.2704	1,189.32	1,981	53
	PM	52.2676	434.55	663	20
	PDM	52.2867	688.61	1,276	4
	ACCPM	52.2677	3,441.67	14	11
2.0	FD	72.6471	1,843.18	3,093	58
	PM	72.6433	471.17	688	17
	PDM	72.6846	1,168.62	2,097	6
	ACCPM	72.6433	3,186.16	14	11
2.5	FD	94.8886	2,733.1	4,443	60
	PM	94.8835	558.02	741	17
	PDM	94.9672	1,114.79	2,022	5
	ACCPM	94.8837	3,276.89	13	11
3.0	FD	119.3122	3,824.44	6,118	63
	PM	119.3057	501.78	691	20
	PDM	119.4434	1,729.36	3,054	7
	ACCPM	119.3060	3,544.01	13	11

networks. The first two are given by the CNET and the eight other are randomly generated networks. The network generator which has been already used to validate ACCPM on nonlinear multicommodity flow problems, is described in more details in Goffin et al. (1997). The problems in Table 4 differ in the size and the structure of the network (they are listed in increasing order of the number of arcs). As for the 106 nodes-904 arcs network, there is a great diversity in the methods. A significant observation from this table is that problem dimension is far to be the most determinant parameter to explain efficiency of the methods. While PM seems to be faster to reach its stopping criterion, it may happen that it converges slower (see results on the 300 nodes-2,000 arcs network). ACCPM behaves better on these randomly generated problems. PDM still provides solutions with less path dispersion. Recall that its performance can be improved by a best choice of the penalty parameter λ (see Figure 4). The slow convergence of FD is also observed in Table 4 but it sometimes converges faster than PDM as already observed in Mahey et al. (1998).

Finally, we have generated ten instances of a 300 nodes-2,000 arcs network with 1,000 commodities. Table 5 collects statistical information obtained on these problems about the CPU time, the number of iterations, and the path dispersion. It confirms some general conclusions drawn from Table 4. The largest standard deviation in CPU time and numbers of iterations occurs with PM while ACCPM is the less variable. The other two methods have intermediate variability though FD and PDM are closer to ACCPM than PM.

These limited numerical experiments do not allow us to draw definitive conclusions on the comparative behavior of the tested algorithms. Our aim was to compare different aspects of the convergence of some among the most efficient methods for the message routing problem. Indeed, the above considerations indicate that the choice of the most adequate algorithm for decision makers will be oriented by analyzing the tradeoff between precision and the different criteria discussed above for a specific application.

Table 4 Results on Some Randomly Generated Networks

Network							
# Nodes	# Arcs	*K*		Delay	CPU Time	Number of Iter.	Path Dispersion
19	68	55	FD	34.5839	2.25	1,792	5
			PM	34.5815	0.07	44	4
			PDM	34.5817	0.74	477	4
			ACCPM	34.5815	0.74	11	5
21	68	420	FD	68.8435	2.32	594	8
			PM	68.8389	1.13	94	4
			PDM	68.8389	7.79	980	4
			ACCPM	68.8390	4.0	13	6
60	280	100	FD	53.0840	27.41	1,397	8
			PM	53.0807	0.43	36	5
			PDM	53.0809	7.65	644	5
			ACCPM	53.0808	6.74	12	8
100	600	200	FD	84.9723	82.99	988	8
			PM	84.9674	35.13	708	6
			PDM	84.9675	125.86	2,507	4
			ACCPM	84.9676	34.32	15	8
100	800	500	FD	139.1048	222.23	2,151	9
			PM	139.0965	8.26	92	7
			PDM	139.0968	156.53	2,122	7
			ACCPM	139.0970	72.94	13	8
150	1,000	2,000	FD	258.4942	932.98	2,452	8
			PM	258.4781	18.23	58	5
			PDM	258.4781	756.09	3,007	5
			ACCPM	258.4790	613.91	14	8
200	1,200	1,000	FD	291.6393	1,819.25	2,548	15
			PM	291.6221	52.38	123	9
			PDM	291.6223	1,523.08	3,601	7
			ACCPM	291.6241	383.93	13	9
200	1,600	2,500	FD	339.5748	1,832.65	2,029	20
			PM	339.5553	78.04	143	9
			PDM	339.5553	4,890.37	8,942	8
			ACCPM	339.5573	1,164.47	14	9
300	2,000	1,000	FD	304.4062	3,397.54	1,346	25
			PM	304.3894	13,110.5	9,949	13
			PDM	304.3922	5,387.12	3,739	8
			ACCPM	304.3902	779.11	15	11
300	2,000	2,000	FD	474.8706	6,167.79	2,340	12
			PM	474.7893	292.1	204	10
			PDM	474.7893	5,101.6	3,483	5
			ACCPM	474.7899	3,073.26	14	9

6. Conclusion

The difficulty in solving convex multicommodity flow problem stems from the coupling constraints and from the fact that real-life instances have very large dimensions. This difficulty is overcome by decomposition approaches which separate the is-

Table 5 Statistic on Ten Instances of a 300 Nodes-2,000 Arcs Generated Network with 1,000 Commodities

		Average	Min.	Max.	Standard Deviation	Norm. Stand. Dev.
CPU Time	FD	4,608.2	2,855	6,500	1,231.7	0.267
	PM	3,398.2	389	13,111	4,294.7	1.264
	PDM	5,188.7	3,547	6,869	1,134.4	0.219
	ACCPM	848.4	612	1,441	243.4	0.287
Number of Iterations	FD	1,920.2	1,346	2,550	380.6	0.198
	PM	2,488.6	289	9,949	3,103.4	1.247
	PDM	3,942.7	2,857	5,399	890.5	0.226
	ACCPM	14.6	14	16	0.70	0.048
Path Dispersion	FD	15.4	9	25	4.74	0.308
	PM	10.3	7	13	2.26	0.219
	PDM	9.2	6	11	1.81	0.197
	ACCPM	11.4	11	12	0.52	0.046

sues of finding flows for the individual commodities and coordinating the flows among the commodities to match the joint capacity constraints. In this paper, we reviewed the main solution techniques. Comparing algorithms from reported results in the literature is difficult since the authors perform their experiments on different computers, code their method in different programming languages, and run different test problems. However, our experiments on a real-life instance (and some randomly generated ones) of the important message routing (flow assignment) problem in communication networks give a fair idea of the relative performance of the four main algorithms described in the paper. In particular, it appears that the user may discriminate algorithms in view of a particular application according to his/her tradeoff between precision and the different characteristics discussed above.[6]

[6] We are indebted to Professors D. P. Bertsekas, MIT, and Larry J. LeBlanc, Vanderbilt University, for kindly providing respectively the PM code (Bertsekas et al. 1984) and the FD code. We thank J. Gondzio, University of Edinburgh, for his help on running the ACCPM software. We are also grateful to J. Chifflet, CNET France Telecom, for his help and comments on this work. We appreciate very much the comments and suggestions received from K. Kiwiel, Polish Academy of Sciences. Finally, we would like to thank the anonymous referees for suggesting a variety of improvements and corrections to this paper. The research of the first two authors is partially supported by CNET-France Telecom (Grant #96-1B021). The research of the third author is partially supported by the Fonds National Suisse (Grant #12-42503.94).

References

Ahuja, R. V., T. L. Magnanti, J. B. Orlin. 1993. *Network Flows: Theory, Algorithms, and Applications.* Prentice Hall, Englewood Cliffs, NJ.

Assad, A. 1978. Multicommodity network flows—a survey. *Networks* **8** 37–91.

Authie, G. 1987. *Contribution à l'optimisation de flots dans les réseaux. Un multiprocesseur expérimental pour l'étude des itérations asynchrones.* Thèse de doctorat d'état, LAAS, Toulouse, France.

Bertsekas, D. P. 1982. Projected Newton methods for optimization problems with simple constraints. *Siam J. Control Optim.* **20** 221–246.

——, E. M. Gafni. 1983. Projected Newton methods and optimization of multicommodity flows. *IEEE Trans. Automat. Contr.* **AC-28** 1090–1096.

——, R. Gendron, W. K. Tsai. 1984. Implementation of an optimal multicommodity network flow algorithm based on gradient projection and a path flow formulation. Report LIDS-P-1364. MIT Laboratory for Information and Decision Systems, Cambridge, MA.

——, R. G. Gallager. 1987. *Data Networks.* Prentice-Hall, Englewood Cliffs, NJ.

——, J. N. Tsitsiklis. 1989. *Parallel and Distributed Computation.* Prentice-Hall, Englewood Cliffs, NJ.

Cantor, D., M. Gerla. 1974. Optimal routing in a packet-switched computer network. *IEEE Trans. Comput.* **C-23** (10) 1062–1069.

Cheney, W., A. Goldstein. 1959. Newton's method for convex programming and Chebishev approximation. *Num. Math.* **1** (5) 253–268.

Chifflet, J., P. Mahey, V. Reynier. 1994. Proximal decomposition for multicommodity flow problems with convex costs. *Telecommunication Systems* **3** 1–10.

Choi, I. C., D. Goldfarb. 1990. Solving multicommodity network flow problems by an interior point method. F. Coleman, Yuying Li eds. *Large-Scale Numerical Optimization*. SIAM 58–69.

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

du Merle, O., J. L. Goffin, J. P. Vial. 1998. On improvements to the analytic center cutting plane method. *Comput. Optim. Applications* **11** 37–52.

Eckstein, J. E. 1989. Splitting methods for monotone operators with applications to parallel optimization. Ph.D. Thesis, Department of Civil Engineering, Massachusetts Institute of Technology, Cambridge, MA.

——, M. Fukushima. 1993. Some reformulations and applications of the alternating direction method of multipliers. *Large Scale Optimization: State of the Art*. Kluwer Academic Publishers B.V., The Netherlands, 119–138.

——. 1994. Parallel alternating direction multiplier decomposition of convex programs. *J. Optim. Theory and Applications* **80** 39–62.

Farvolden, J. M., W. B. Powell, I. J. Lustig. 1993. A primal partitioning solution for the arc-chain formulation of a multicommodity network flow problem. *Oper. Res.* **41** (4) 669–693.

Florian, M., J. Guelat, H. Spiess. 1987. An efficient implementation of the PARTAN variant of the linear approximation method for network equilibrium problem. *Networks* **17** 319–339.

Ford, L. R., D. R. Fulkerson. 1962. *Flows in Networks*. Princeton University Press, Princeton, NJ.

Franck, M., P. Wolfe. 1956. An algorithm for quadratic programming. *Naval Res. Logist. Quart.* **3** 95–110.

Fratta, L., M. Gerla, L. Kleinrock. 1973. The flow deviation method: An approach to store-and-forward communication network design. *Networks*, **3** 97–133.

Fukushima, M. 1984. A nonsmooth optimization approach to nonlinear multicommodity network flow problems. *J. Oper. Res. Soc. Japan* **27** (2) 151–177.

Gabrel, V., M. Minoux. 1996. Large scale LP relaxations for minimum cost multicommodity flow problems with step increasing cost functions and computational results. Technical Report 96/17, Laboratoire MASI, Université Paris 6, France.

de Ghellinck, G., J.-P. Vial. 1986. A polynomial Newton method for linear programming. *Algorithmica* **1** 425–453.

Goffin, J.-L., J. Gondzio, R. Sarkissian, J. P. Vial. 1997. Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. *Math. Programming* **76** 131–154.

——, A. Haurie, J.-P. Vial. 1992. Decomposition and nondifferentiable optimization with the projective algorithm. *Management Sci.* **38** (2) 284–302.

Goldberg, A. V., R. E. Tarjan. 1989. Finding minimum cost circulations by canceling negatives cycles. *J. Assoc. Comput. Mach.* **36** (4) 873–886.

Gondran, M., M. Minoux. 1979. *Graphes et Algorithmes*. Eyrolles, France.

Gondzio, J., R. Sarkissian, J.-P. Vial. 1997. Using an interior point method for the master problem in a decomposition approach. *European J. Oper. Res.* **101** (3) 577–587.

Hossein, P. A., D. P. Bertsekas, P. Tseng. 1987. Relaxation methods for network flow problems with convex arc costs. *SIAM. J. Control Optim.* **5** (25) 1219–1243.

Hu, T. C. 1963. Multi-commodity network flows. *Oper. Res* **11** 344–360.

Ibaraki, S., M. Fukushima. 1994. Primal-dual proximal point algorithm for multicommodity network flow problems. *J. Oper. Res. Soc. Japan* **37** (4) 297–309.

Idrissi, H., O. Lefebvre, C. Michelot. 1989. Applications and numerical convergence of the partial inverse method. *Optimization*, Lecture Notes in Math. 1405. Springer Verlag, Germany, 39–54.

Jones, K. L., I. J. Lustig, J. M. Farvolden, W. B. Powell. 1993. Multicommodity network flows: The impact of formulation on decomposition. *Math. Programming* **62** 95–117.

Karzanov, A. V., S. T. McCormick. 1997. Polynomial methods for separable convex optimization in totally unimodular linear spaces with applications to circulations and co-circulations in networks. *SIAM J. Comput.* **26** 1245–1275.

Kelley, J. E. 1960. The cutting plane method for solving convex programs. *J. Soc. Indust. Appl. Math.* **8** 703–712.

Kennington, J. F. 1978. A survey of linear cost multicommodity network flows. *Oper. Res.* **2** (26) 209–236.

Kleinrock, L. 1964. *Communication Nets: Stochastic Message Flow and Delay*. McGraw-Hill, New York.

Lasdon, L. S. 1970. *Optimization Theory for Large Systems*. Macmillan, New York.

LeBlanc, L. 1973. Mathematical programming algorithms for large scale network equilibrium and network design problems. Ph.D. Dissertation, IE/MS Dept., Northwestern University, Evanston IL.

——, E. Morlok, W. Pierskalla. 1975. An efficient approach to solving the road network equilibrium traffic assignment problem. *Trans. Res.* **9** 309–318.

——, R. Helgason, D. E. Boyce. 1985. Improved efficiency of the Frank-Wolfe algorithm for convex network programs. *Transportation Sci.* **19** 445–462.

Lemaréchal, C. 1974. Méthodes de sous-gradients. *Bulletin de la Direction des Etudes et Recherches EDF* **C** (2) 5–14.

Mahey, P., S. Oualibouch, D. T. Pham. 1995. Proximal decomposition on the graph of a maximal monotone operator. *SIAM J. Optim.* **5** (2) 454–466.

——, A. Ouorou, L. LeBlanc, J. Chifflet. 1998. A new proximal decomposition algorithm for routing in telecommunication networks. *Networks* **31** 227–238.

Medhi, D. 1994. Bundle-based decomposition for large scale convex optimization: Error estimate and application block-angular linear programs. *Math. Programming* **66** (1) 79–102.

Minoux, M. 1989. Network synthesis and optimum network design problems: Models, solution methods and applications. *Networks*, Vol. 19, John Wiley & Sons, New York, 313–360.

Nagamochi, H. 1988. Studies on multicommodity flows in directed networks. thesis, Eng. Dr. thesis, Kyoto University, Japan.

Ouorou, A. 1995. Décomposition proximale des problèmes de multiflot à critère convexe. Application aux problèmes de routage dans les réseaux de communication. Thèse Doctorale, Université Blaise Pascal, Clermont-Ferrand, France.

Ouorou, A., P. Mahey. 1996. Minimum mean cycles canceling method for nonlinear multicommodity flow problems. *European J. Oper. Res.* Forthcoming.

Patriksson, M. 1994. *The traffic assignment problem: models and methods.* Topics in Transportation, VSP, Utrecht, The Netherlands.

Pinar, M. C., S. Zenios. 1992. Parallel decomposition of multicommodity network flows using a linear quadratic penalty algorithm. *ORSA J. Computing* **4** (3) 235–249.

——, S. A. Zenios. 1993. A comparative study of parallel decompositions for multicommodity flow problems. *Parallel Algorithms and Applications* **1** 255–271.

Rockafellar, R. T. 1976. Monotone operators and the proximal point algorithm. *SIAM. J. Control and Optim.* **14** 877–898.

——. 1970. *Convex Analysis,* Princeton University Press, Princeton, NJ.

——. 1976. Augmented Lagrangians and applications of the proximal point algorithm in convex programming. *Math. Oper. Res.* **1** 97–116.

Schultz, G. L., R. R. Meyer. 1991. An interior point method for block angular optimization. *SIAM J. Optim.* **1** (4) 583–602.

Schwartz, M., C. Cheung. 1976. The gradient projection algorithm for multiple routing in message-swicthed networks. *IEEE Trans. on Communications* **COM-24** 449–456.

——, T. E. Stern. 1980. Routing techniques used in computer communication networks. *IEEE Trans. on Communications* **COM-28** 539–552.

Spingarn, J. E. 1983. Partial inverse of a monotone operator. *Appl. Math. Optim.* **10** 247–265.

——. 1985. Applications of the method of partial inverse to convex progr decomposition, *Math. Programming* **32** 199–223.

Stern, T. E. 1977. A class of decentralized routing algorithms using relaxation. *IEEE Trans. Communications* **COM-25** 1092–1102.

Tseng, P. 1990. Dual ascent methods for problems with strictly convex costs and linear constraints: A unified approach, *SIAM J. Control and Optim.* **28** 214–242.

——, D. P. Bertsekas. 1987. Relaxation methods for problems with strictly convex separable arc costs and linear constraints. *Math. Programming* **38** 303–321.

Von Hohenbalken, B. 1977. Simplicial decomposition in nonlinear programming algorithms. *Math. Programming* **13** 49–68.

Wolfe, P. 1975. A method of conjugate subgradients for minimizing nondifferentiable functions, *Nondifferentiable Optimization, Math. Programming Study* **3** 145–173.

Zangwill, W. 1969. *Nonlinear Programming: A Unified Approach,* Prentice-Hall, Englewood Cliffs, NJ.

Zenios, S. A., Y. Censor. 1991. Massively parallel row-action algorithms for some nonlinear transportation problems. *SIAM J. Optim.* **1** 373–400.

Accepted by Thomas M. Liebling; received June 1997. This paper has been with the authors 11 months for 2 revisions.