# Interior Point Methods With Decomposition For Multicommodity Flow Problems

J. Frédéric Bonnans[*], Mounir Haddou[†], Abdel Lisser[‡] and Raja Rébaï[§]

Thème 4 — Simulation et optimisation
de systèmes complexes
Projet Mocoa

Rapport de recherche n°3852 — Janvier 2000 — 23 pages

**Abstract:** This paper introduces an approach by decomposition of an interior point method for solving multicommodity flow problems. First, we present this approach in the general framework of coupling constraints problems. Next, we propose to specialize the algorithm to the linear multicommodity network-flow problems. We expose this specialization using the node-arc formulation. Then, we focus on the arc-path formulation and we propose decomposition method witch incorporates the interior point method into the Dantzig-Wolfe decomposition technique. The numerical results show the superiority of this last formulation. Finally, we report some numerical results obtained by testing these algorithms with data from the France-Telecom Paris district transmission network.

**Key-words:** Large-scale linear programming, interior point methods, decomposition, multicommodity network flow models, predictor corrector algorithm.

*(Résumé : tsvp)*

Rapport France Telecom/BD/NT/Cnet/6573, Marché CNET-INRIA 961B115.

[*] Email: Frederic.Bonnans@inria.fr
[†] INRIA and Dept. Math, Université d'Orléans, France. Email: haddou@labomath.univ-orleans.fr
[‡] CNET, France Telecom, 38-40 Rue du Général Leclerc, F-92131 Issy Les Moulineaux, France. Email: abdel.lisser@cnet.francetelecom.fr
[§] Email: Raja.Rebai@inria.fr

# Méthodes de points intérieurs avec décomposition pour la résolution de problèmes de multiflot

**Résumé :** Cet article introduit une approche par décomposition d'une méthode de points intérieurs pour la résolution d'un problème de multiflot. Nous présentons d'abord cette approche dans le cadre général de problèmes avec contraintes couplantes. Nous proposons ensuite de spécialiser l'algorithme aux problèmes de multiflot linéaire. Nous exposons cette spécialisation en utilisant la formulation nœuds-arcs. Nous nous concentrons ensuite sur la formulations arcs-chemins et nous proposons une méthode de décomposition qui incorpore la méthode de points intérieurs dans la technique de décomposition de Dantzig-Wolfe. Les résultats numériques montrent la supériorité du dernier algorithme. Enfin, nous présentons des résultats numériques obtenus en testant ces algorithmes sur des problèmes fournis par le CNET.

**Mots-clé :** Programmation linéaire de grande taille, méthodes de points intérieurs, décomposition, problèmes de multiflots, algorithme prédicteur correcteur.

## 1 Introduction

In this paper, we are concerned with the Least cost Multicommodity Network-flow Problem (LMNP), which is the most studied multicommodity problem. This problem consists in the determination of an optimal routing of the traffic requirements when designing a network.
The first study of multicommodity network-flow problems was conducted by Ford and Fulkerson in 1958 [FF62]. Their work inspired Dantzig and lead to the price decomposition [Dan61]. Recently, in [CL97] Chardaire and Lisser present various approaches based on specialization of the simplex algorithm and interior-point methods for solving non-oriented multicommodity flow problems. Another recent work in the context of interior point algorithms for network flow problems, can be found in [RÓ0], where it is shown that this kind of method may be very effective.
For being competitive, the algorithm must exploit the structure of multicommodity problem to solve efficiently Newton directions systems.
Most of specialized methods attempt to exploit in some way the block structure of the multicommodity problem. The price directive or Dantzig-Wolfe decomposition is regarded as successful approach in [AHKL80] and belongs to the class of cost decomposition approaches for multicommodity flows (see [Fra97], [GGSV97] and [Zen95] for recent variants based on bundle methods, analytic centers, and smooth penalty functions, respectively).
The best complexity bound known for multicommodity problems is provided by an interior point algorithm [KV96], but as yet no efficient implementation has been obtained. The major work in a single iteration of any Interior Point Method (IPM) consists in solving a set of linear equations, the so-called Newton equation system (for details see [AGMX96, BGLS97]). All general purpose IPM codes use direct approach [DER89] to solve the Newton equation system. In [KKR93], Kamath and al. applied a variant of Karmakar's projective algorithm using a preconditioned conjugate gradient (PCG). However, their preconditioner did not take advantage of the multicommodity structure. In [Casar], Castro exploited this structure and presents a specialization of an interior point algorithm to multicommodity flow problems. He uses both a preconditioned conjugate gradient solver, and a sparse Cholesky factorization, to solve a linear system of equations at each iteration of the al! gorithm. Choi and Goldfarb, in [CG90], presented a decomposition scheme similar to the one in [Casar]. But they suggest solving a dense-matrix positive definite linear system that appears during the decomposition stage by means of parallel and vector processing. Some similarities were also found between the decomposition scheme presented by Castro in [Casar] and ours presented in this paper.
The interest of our approach rests in the possibility of solving the arc-path formulation of (LMNP). As will be shown, this formulation is more efficient than the node-arc one.
The linear system to be solved for computing the Newton direction appears, in the case of linear multicommodity network-flow problems, to have the following decoupling property: fixing the value of the coupling variables, we can solve in parallel some equations that coincide with those of the Newton step for the central path associated with the subproblems. Solving the resulting reduced system allows us to compute the Newton direction, while respecting the above mentioned decomposition principle.

This strategy of computing the Newton direction enables our algorithm to solve efficiently large problems. The interior point algorithm, presented in this paper belongs to the path following interior point algorithms family. These algorithms focused the attention of the community because they both reach the optimal complexity known until now, i.e. convergence within $O(\sqrt{n}L)$ iterations, while converging quadratically [Gon92],[KMNY91]. A single iteration of predictor-corrector method needs two solves of the same kind of large, sparse linear system for two different right hand sides. The resolution of these systems represents the main computational burden of the algorithm.
The performance of the predictor-corrector algorithm relies on the efficient solution on those systems.
In the case of linear multicommodity network-flow problems, to have the following decoupling property: fixing the value of the coupling variables, we can solve in parallel some equations that coincide with those of the Newton step for the central path associated with the subproblems.
This paper is organized as follows. Section 2 presents the approach by decomposition of the interior point method in the general framework of coupling constraints problems. Section 3 describes the specialization of the algorithm to the linear multicommodity network-flow problem. First, we present this specialization using the node-arc formulation. Then, we focus on the arc-path formulation and we propose decomposition method witch incorporates the interior point method into the Dantzig-Wolfe decomposition technique. Finally, we report some numerical results in section 4.

**Conventions**
Given a vector $x \in \mathbb{R}^n$ the relation $x > 0$ is equivalent to $x_i > 0,\ i = 1, 2, \ldots, n$, while $x \geq 0$ means $x_i \geq 0,\ i = 1, 2, \ldots, n$. We denote $\mathbb{R}^n_+ = \{x \in \mathbb{R}^n : x \geq 0\}$ and $\mathbb{R}^n_{++} = \{x \in \mathbb{R}^n : x > 0\}$. Given a vector $x$, the corresponding upper case symbol denotes as usual the diagonal matrix $X$ defined by the vector. The symbol $\mathbf{1}$ represents the vector of all ones, with dimension given by the context. We denote component-wise operations on vectors by the usual notations for real numbers. Thus, given two vectors $u, v$ of the same dimension, $uv$, $u/v$, etc. will denote the vectors with components $u_i v_i$, $u_i/v_i$, etc. We always formulate an optimization problem as follows:

$$\left\{ \begin{array}{lll} \underset{x}{\text{Min}} & f(x); & \\ s.t. & g(x) \leq 0, & (\Lambda) \end{array} \right. \tag{1}$$

where $\Lambda$ denotes the Lagrange multipliers associated with the constraint.
We denote the null space and range space of a matrix $A$ by $\mathcal{N}(A)$ and $\mathcal{R}(A)$ respectively.

## 2 Linear Optimization problems with coupling constraints

Linear coupling constraints problems usually have a very large number of variables and constraints and arise in a great variety of applications [DW60, Dan61]. This section is concerned with the resolution of these problems using an interior point method, the predictor-corrector method.

## 2.1 Problem formulation

We consider the so called linear optimization problem with coupling constraints:

$$\begin{cases} \operatorname*{Min}_{x^0,x^1,\ldots,x^K} \sum_{k=0}^{K} (c^k)^T x^k; & \\ \sum_{k=0}^{K} B^k x^k = b^0, & (\lambda^0) \\ A^k x^k = b^k, k = 1, \ldots, K, & (\lambda^k) \\ x^k \geq 0, k = 0, \ldots, K, & (s^k) \end{cases} \tag{2}$$

where

- for all $k \in \{0, \ldots, K\}$ :
  - $x^k$, $c^k$ and $s^k$ (dual variables of $x^k$) are elements of $\mathbb{R}^{n^k}$.
  - $b^k$ and $\lambda^k$ (dual variable of the $k^{th}$ constraint) are elements of $\mathbb{R}^{p^k}$.
  - and $B^k$ is a $p^0 \times n^k$ matrix.
- for all $k \in \{1, \ldots, K\}$: $A^k$ is a $p^k \times n^k$ matrix.

Without the first constraints problem (2) would be separable. Indeed, each $x^k$ is in this case solution of the linear problem

$$\operatorname*{Min}_{z \in \mathbb{R}^{n_k}} (c^k)^T z \; ; \; A^k z = b^k \; ; \; z \geq 0. \tag{3}$$

One recovers the standard form of linear optimization by setting

$$\hat{x} := \begin{pmatrix} x^0 \\ x^1 \\ \vdots \\ x^K \end{pmatrix} \in \mathbb{R}^{n^t}, \quad \hat{c} := \begin{pmatrix} c^0 \\ c^1 \\ \vdots \\ c^K \end{pmatrix} \in \mathbb{R}^{n^t}, \quad \hat{b} := \begin{pmatrix} b^0 \\ b^1 \\ \vdots \\ b^K \end{pmatrix} \in \mathbb{R}^{p^t},$$

$$\text{and the } p^t \times n^t \text{ matrix } \hat{A} = \begin{pmatrix} B^0 & B^1 & \cdots & B^K \\ & A^1 & & \\ & & \ddots & O \\ & O & & A^K \end{pmatrix},$$

where $n^t := \sum_{k=0}^{K} n^k$ and $p^t := \sum_{k=0}^{K} p^k$. We assume that $A^1$, $\ldots$, $A^K$, and $B^0$ have full row rank; it follows that $\hat{A}$ itself has full row rank. Problem (2) is equivalent to the linear optimization problem in standard form

$$\left\{\begin{array}{ll} \mathrm{Min}_{\hat{x}}\, \hat{c}^T \hat{x}; & \\ \hat{A}\hat{x} = \hat{b}, & (\hat{\lambda}) \\ \hat{x} \geq 0, & (\hat{s}) \end{array}\right. \tag{4}$$

The Lagrange multipliers vector $\hat{\lambda}$ ($\in \mathbb{R}^{p^t}$) and $\hat{s}$ ($\in \mathbb{R}^{n^t}$) can be expressed with Lagrange multipliers of problem (2). Indeed, we have

$$\hat{s} := \begin{pmatrix} s^0 \\ s^1 \\ \vdots \\ s^K \end{pmatrix} \quad \text{and} \quad \hat{\lambda} := \begin{pmatrix} \lambda^0 \\ \lambda^1 \\ \vdots \\ \lambda^K \end{pmatrix}.$$

The first bloc of the matrix $\hat{A}$ corresponds to the coupling constraints. Without these constraints, the program (4) would have a diagonal structure of which we might take advantage.

## 2.2 solving linear optimization problems with coupling constraints using a predictor-corrector algorithm

We are concerned with the most popular primal-dual logarithmic barrier method called predictor-corrector method. This name was first used by Mizuno, Todd and Ye in [MTY93], where they proposed for the first time the method considered in this paper for the small neighborhood case. Like them we alternate (single) primal-dual affine-scaling step and (single) primal-dual centering step but using large neighborhoods of the central path [BPR00]. We choose to use this method because of its good theorical and practical properties. Indeed, the predictor-corrector methods converges with an $O(nL)$ iterations bound (when large neighborhoods are used) and has asymptotically a quadratic convergence (see .e.g., Mehrotra [Meh92, Meh93], Ye et al [YGTZ93], Gonzaga and Tapia [GT97a, GT97b], Ye [Ye92] and Luo and Ye [LY94]). For more details about primal-dual predictor-corrector methods, we refer the reader to Bonnans et al [BGLS97], Roos et al [RTV97], Wright [Wri97].
Interior point methods follow the so-called center path (approximately) as a guideline to the optimal set. Subsequently we exploit the structure of the central path equation to obtain a reduced equation. Then, we deduce a decomposition method for computing the Newton directions.

### 2.2.1 Predictor-corrector algorithm

Predictor-corrector algorithms follow the central path for solving (4). The equations of the central path associated with the linear optimization problem in its standard form (4) are

$$\left\{\begin{array}{lll} \hat{x}\hat{s} & = & \mu\mathbf{1}, \\ \hat{A}\hat{x} & = & \hat{b}, \\ c + \hat{A}^T\hat{\lambda} & = & \hat{s}, \end{array}\right. \tag{5}$$

where $(\hat{x}, \hat{s}, \hat{\lambda})$ belongs to $\mathbb{R}^{n^t}_{++} \times \mathbb{R}^{n^t}_{++} \times \mathbb{R}^{p^t}$.
At each iteration, the algorithm compute an affine and centering Newton directions of the following form:

$$\begin{cases} \hat{s}\hat{u} + \hat{x}\hat{v} &= \hat{f}, \\ \hat{A}\hat{u} &= 0, \\ \hat{A}^T\delta &= \hat{v}, \end{cases} \tag{6}$$

where $(\hat{u}, \hat{v}, \delta)$ denotes the step associated with $(\hat{x}, \hat{s}, \hat{\lambda})$, and the right hand side $\hat{f}$ depends on the algorithm.
We state in the following a predictor corrector algorithm in large neighborhoods (**PCL**), in which the set $\mathcal{N}_\nu$ denotes a large neighborhood defined by:

$$\mathcal{N}_\nu := \Big\{(\hat{x}, \hat{s}, \hat{\lambda}, \mu) \in \mathbb{R}^{n^t}_{++} \times \mathbb{R}^{n^t}_{++} \times \mathbb{R}^{p^t} \times \mathbb{R}_{++}; \hat{A}\hat{x} = \hat{b},\ c + \hat{A}^T\hat{\lambda} = \hat{s} \text{ and } \nu\mathbf{1} \le \frac{xs}{\mu} \le \nu^{-1}\mathbf{1},\ \mu \le \mu_0\Big\}.$$

The algorithm is as follows:

```
Algorithm PCL Data: $\mu_\infty > 0,\ \nu \in ]0, 1/2],\ (\hat{x}^0, \hat{s}^0, \hat{s}\mu_0) \in \mathcal{N}_\nu.\ j := 0$
  REPEAT
    • $\hat{x} := \hat{x}^j,\ \hat{s} := \hat{s}^j,\ \mu := \mu_j$ ;
    • Corrector step: Compute $(\hat{u}^c, \hat{v}^c)$ solution of (6) with $\hat{f} = -\hat{x}\hat{s} + \mu\mathbf{1}$.
      $\hat{x}(\theta) := \hat{x} + \theta\hat{u}^c,\ \hat{s}(\theta) := \hat{s} + \theta\hat{v}^c$.
      Compute $\theta^c \in ]0, 1]$ such that $(\hat{x}(\theta^c), \hat{s}(\theta^c), \mu) \in \mathcal{N}_\nu$.
      $\hat{x} := \hat{x}(\theta^c),\ \hat{s} := \hat{s}(\theta^c)$.
    • Predictor step: Set $\hat{f} = -\hat{x}\hat{s}$. Compute $(\hat{u}^a, \hat{v}^a)$ solution of (6).
      $\hat{x}(\theta) := \hat{x} + \theta\hat{u}^a,\ \hat{s}(\theta) := \hat{s} + \theta\hat{v}^a,\ \mu(\theta) := (1-\theta)\mu$ .
      Compute $\theta^a$, the largest value in $]0, 1[$ such that
      $(\hat{x}(\theta), \hat{s}(\theta), \mu(\theta)) \in \mathcal{N}_\nu,\ \forall\ \theta \le \theta^a$.
      $\hat{x}^{j+1} := \hat{x}(\theta^a),\ \hat{s}^{j+1} := \hat{s}(\theta^a),\ \mu_{j+1} := (1-\theta^a)\mu_j$.
    • $j := j + 1$ .
  UNTIL $\mu_j < \mu_\infty$.
```

We choose the step-size for the centering displacement is as follows: starting from a unit step, we divide the step by two until the new point belongs to the large neighborhood. This is a rather rough linear search. Because it proved to be efficient, we content of it. The algorithm finds a point such that $\mu_k \le \mu_\infty$ in at most $O(nL)$ iterations [BPR00].

### 2.2.2 The central path

We rewrite the system (5) as follows:

$$\left\{\begin{array}{lcl} x^0 s^0 & = & \mu \mathbf{1}, \\ \sum_{k=0}^{K} B^k x^k & = & b^0, \\ c^0 + (B^0)^T \lambda^0 & = & s^0, \end{array}\right. \tag{7}$$

and

$$k \in \{1, \dots, K\} \left\{\begin{array}{lcl} x^k s^k & = & \mu \mathbf{1}, \\ A^k x^k & = & b^k, \\ c^k + (B^k)^T \lambda^0 + (A^k)^T \lambda^k & = & s^k. \end{array}\right. \tag{8}$$

With a value of $\lambda^0$ (for which we assume that each problem (8) has an interior feasible solution realizing (7) is associated some uniquely defined $(x^k(\lambda^0), s^k(\lambda^0), \lambda^k(\lambda^0))$, $k = 1, \dots, K$, solution of the local problem (8). This local problem may be interpreted as the equation of the central path associated with the local problem (3) where the unitary cost vector is $c^k + (B^k)^T \lambda^0$.

We may write the central path equation in the *reduced form*:

$$\left\{\begin{array}{lcl} x^0 s^0 & = & \mu \mathbf{1}, \\ \sum_{i=0}^{K} B^k x^k(\lambda^0) & = & b^0, \\ c^0 + B^0 \lambda^0 & = & s^0. \end{array}\right. \tag{9}$$

Predictor-corrector algorithm follows the central path using Newton directions on the equation of the central path. We apply now similar considerations for computing the Newton step.

## 2.3 The Newton Step

The system (6) may be written as

$$\left\{\begin{array}{lcl} s^0 u^0 + x^0 v^0 & = & f^0, \\ \sum_{k=0}^{K} B^k u^k & = & 0, \\ (B^0)^T \delta^0 & = & v^0, \\ k \in \{1, \dots, K\} \left\{\begin{array}{lcl} s^k u^k + x^k v^k & = & f^k, \\ A^k u^k & = & 0, \\ (A^k)^T \delta^k + (B^k)^T \delta^0 & = & v^k. \end{array}\right. & & \end{array}\right. \tag{10}$$

We can solve this linear system as follows. For a given value of $\delta^0$ we can compute in parallel the values of $(u^k, v^k, \delta^k)$, $k = 1, \dots, K$ that satisfy the three last equations.

Let us call $u^k(\delta^0)$, $v^k(\delta^0)$, $\delta^k(\delta^0)$ the solution of the $k^{th}$ local problem (8).
Note that $u^k(\delta^0)$, $v^k(\delta^0)$ and $\delta^k(\delta^0)$ are affine functions of $\delta^0$.
Computing the Newton step is then equivalent to solving the reduced linear system

$$\left\{\begin{array}{lcl} s^0u^0 + x^0v^0 & = & f^0, \\ \sum_{k=0}^{K} B^k u^k(\delta^0) & = & 0, \\ (B^0)^T\delta^0 & = & v^0. \end{array}\right. \tag{11}$$

which can be interpreted as the Newton step equation of (7) in the feasible case. Once $\delta^0$ is computed, the resolution of the $K$ systems

$$\left\{\begin{array}{lcl} \text{(i) } s^ku^k + x^kv^k & = & f^k, \\ \text{(ii) } A^ku^k & = & 0, \quad k = 1, \dots, K, \\ \text{(iii) } (A^k)^T\delta^k + (B^k)^T\delta^0 & = & v^k, \end{array}\right. \tag{12}$$

may be done in parallel. For each $k$, this is equivalent to the computation of the Newton direction associated to a single commodity minimum network flow problem.

## 2.4 Resolution of the reduced system

Let us see how to solve the reduced system (11). For each $k \in \{1, \dots, K\}$, we consider the system (12) and we define in a classical way the scaled variables and operators :

$$\bar{f}^k := f^k/(\sqrt{x^ks^k}), \quad d^k := \sqrt{x^k/s^k}, \quad \bar{u}^k := (d^k)^{-1}u^k,$$
$$\bar{v}^k := d^kv^k, \quad \bar{A}^k := A^kD^k, \text{ and } \bar{B}^k := B^kD^k.$$

After scaling system (12), we obtain the equivalent relation :

$$k \in \{1, \dots, K\} \left\{\begin{array}{lcl} \text{(i) } \bar{u}^k + \bar{v}^k & = & \bar{f}^k, \\ \text{(ii) } \bar{A}^k\bar{u}^k & = & 0, \\ \text{(iii) } (\bar{A}^k)^T\delta^k + (\bar{B}^k)^T\delta^0 & = & \bar{v}^k. \end{array}\right. \tag{13}$$

Multiplying equation (iii.13) by $\bar{A}^k$ on the left, and using $\bar{A}^k\bar{v}^k = \bar{A}^k\bar{f}^k$, we may express $\delta^k$ as a function of $\delta^0$ :

$$\bar{A}^k(\bar{A}^k)^T\delta^k = -\bar{A}^k(\bar{B}^k)^T\delta^0 + \bar{A}^k\bar{f}^k. \tag{14}$$

$\bar{A}^k(\bar{A}^k)^T$ is invertible because $A^k$ has full row rank.
We associate with each $k \in \{1, \dots, K\}$, the $p^k$ order invertible symmetric matrix $N^k$ defined by: $N^k = \bar{A}^k(\bar{A}^k)^T$.
We introduce also for each $k$ the $p^k \times p^0$ matrix $M^k_\delta$ and the $\mathbb{R}^{p^k}$-vector $h^k_\delta$ defined respectively by:

$$M^k_\delta = -(N^k)^{-1}\bar{A}^k(\bar{B}^k)^T \text{ and } h^k_\delta = (N^k)^{-1}\bar{A}^k\bar{f}^k. \tag{15}$$

We have by (14) and (15):

$$\delta^k = M_\delta^k \delta^0 + h_\delta^k. \tag{16}$$

Combining with equations (iii.13) and (16), we obtain that:

$$\bar{v}^k = M_v^k \delta^0 + h_v^k, \tag{17}$$

where

- $M_v^k$ is a $n^k \times p^0$ matrix defined by: $M_v^k = (\bar{B}^k)^T + (\bar{A}^k)^T M_\delta^k$.
- $h_v^k$ is a $\mathbb{R}^{n^k}$-vector defined by: $h_v^k = (\bar{A}^k)^T h_\delta^k$.

From the equations (17) and (i.13), we deduce:

$$u^k = M^k \delta^0 + h^k, \tag{18}$$

where

- $M^k$ is a $n^k \times p^0$ matrix defined by: $M^k = -D^k M_v^k$.
- $h_u^k$ is a $\mathbb{R}^{n^k}$-vector defined by: $h^k = D^k(\bar{f}^k - h_v^k)$.

We also express $u^0$ as a function of $\delta^0$ by using the first and third equations of the system (11). We obtain then:

$$u^0 = M^0 \delta^0 + h^0, \tag{19}$$

where

- $M^0$ is a $n^0 \times p^0$ matrix defined by: $M^0 = -D^0(B^0)^T$.
- $h^0$ is a $\mathbb{R}^{n^0}$-vector defined by: $h^0 = f^0/s^0$.

From (18) and (19), the reduced system (11) gives the following $p$-order linear system:

$$M\delta^0 = h, \tag{20}$$

where

- $M$ is a $p^0$ order invertible matrix defined by: $M = \sum_{k=0}^{K} B^k M^k$ .
- $h$ is a $\mathbb{R}^{p^0}$-vector defined by: $h = -\sum_{k=0}^{K} B^k h^k$.

Solving the reduced system (11) amounts to solving system (20).

**Remark 1** *At every iteration of the predictor-corrector algorithm, two different systems of equations (20) have to be solved by computing explicitly the matrices $M^k, k = 0, \ldots, K$.*
*It is possible to solve the reduced system (11) by an iterative algorithm without expliciting those matrixes. In this case we obtain an approximate solution of the linear system (20).*
*In [BPR00], the authors show that the infeasible predictor-corrector algorithm in large neighborhoods remains fast under a relative perturbation of the right-hand-side of the order of* $25\%$*.*
*That proves the robustness of the algorithm when the Newton direction is computed approximately.*

If we look more into the details of the matrix $M$, we check upon the following properties:

**Lemma 1** *The matrix $M$ is symmetric negative definite.*

**Proof 1** *Without loss of generality, let us suppose that $f^k = 0$ and $x^k = s^k = \mathbf{1}$, for $k = 0, \ldots, K$. Indeed, we are looking for matrix properties and neither the right hand side term nor the positive diagonal scaling matrix has a consequence on theses properties. Using these hypothesis and system (10), obtain*

$$\left\{ \begin{array}{lcl} B^0 M^0 & = & -B^0 (B^0)^T, \\ B^k M^k & = & -B^k [I(p_k) - (A^k)^T (A^k (A^k)^T)^{-1} A^k] (B^k)^T, k = 1, \ldots, K. \end{array} \right. \tag{21}$$

*The system (21) proves that each $B^i M^i$ is symmetric for $i \in \{0, 1, \ldots, K\}$ .*
*Since $M = \sum_{k=0}^{K} B^k M^k$ (20), $M$ is also symmetric.*
*Let us check that $M$ is negative definite.*
*We recall that $B^0$ has full row rank. Hence $B^0 M^0$ is negative definite.*
*To complete the proof, it suffice to show that for each $k \in \{1, \ldots, K\}$, $[I(p_k) - (A^k)^T (A^k (A^k)^T)^{-1} A^k]$ is positive semi-definite.*
*For $k \in \{1, \ldots, K\}$ and all $y^k \in \mathbb{R}^{p_k}$, we have ${y^k}^T [I(p_k) - (A^k)^T (A^k (A^k)^T)^{-1} A^k] y^k = \|\tilde{y^k}\|^2$ where $\tilde{y^k}$ is the projection of $y^k$ on $\mathcal{N}(A^k)$.*
*The conclusion follows.*

**Remark 2** *Thanks to the good properties of the linear system (20) one can use the conjugate gradient method for solving the reduced system as an iterative method. In this study, we propose a generic predictor-corrector algorithm for solving linear optimization problems with coupling constraints. For more details about using preconditioned conjugate gradient to solve an example of those problem, we refer the reader to [Casar]. The reduced system matrix $M$ is known as the Schur complement [Casar]. Choi and Goldfarb in [CG90] and also Castro in [Casar] state that the Schur complement becomes completely dense. In order to circumvent this difficulty, Castro propose to solve the reduced system through a preconditioned conjugate gradient. The preconditioner that he proposed consist of using the inverse of $M$. However in the specialization proposed in [Casar], Castro takes advantage only of* *node-arc linear multicommodity problem structure.*

We describe below only one Newton step of the predictor corrector algorithm PCL.

**Algorithm PCLMNP**

- for $k \in \{1,\dots,K\}$: Compute the matrix $M^k$ and the vector $h^k$, using (16),(17) and (18):
    - $M^k := -D^k(\bar{B}^k)^T + (\bar{A}^k)^T(\bar{A}^k(\bar{A}^k)^T)^{-1}\bar{A}^k(\bar{B}^k)^T)$
    - $h^k := D^k(\bar{f}^k - (\bar{A}^k)^T(\bar{A}^k(\bar{A}^k)^T)^{-1}\bar{A}^k\bar{f}^k)$.
- Compute the matrix $M^0$ and the vector $h^0$, using (19)
    - $M^0 := -D^0(B^0)^T$.
    - $h^0 := f^0/s^0$.
- Solve the reduced system (20).

  Solve the linear system $M\delta^0 = h$, where
    - $M := \sum_{k=0}^{K} B^k M^k$,
    - $h := -\sum_{k=0}^{K} B^k h^k$.

  $v^0 := \delta^0$ and $u^0 := (f^0 - x^0\delta^0)/s^0$.
- for $k = 1,\dots,K$
    - compute $(u^k, v^k, \delta^k)$ solution of the system (12).

## 3 Linear multicommodity network-flow problem

The Multicommodity network problems appear when different commodities share a common network. These problems are generally very difficult to solve, not because of their large scale, but also because it is not easy to handle the conflicts between several commodities on the same network [CMR94]. There are numerous models in applied optimization that involve multicommodity systems [Ass78], [GGSV97], [LSV95a], [LSV95b]. The Linear multicommodity network-flow problem is the most often studied problem in multicommodity optimization [Ken78],[AMO93] and [CL97].
This problem consists of the determination of the most economical way of using the available transmission capacities in order to route a traffic matrix through the network.
The best complexity bound known for multicommodity problems is provided by an interior

point algorithm [KV96], though, as yet, no efficient implementation had been obtained. A variant of Karmarkar's projective algorithm is applied in [KKR93], using a preconditioned conjugate gradient (PCG) solver. Contrarily to the preconditioner presented in [KKR93] that did not take advantage of the multicommodity structure, the one proposed in [CG90] exploit this structure. Recently Castro presented in [Casar] a specialization of an interior point algorithm to multicommodity flow problems. He uses both a preconditioned conjugate gradient solver, and a sparse Cholesky factorization, to solve a linear system of equations at each iteration of the algorithm.

## 3.1 Framework

Let $G(V,E)$ be a directed graph, where $V$ is a set of $p$ nodes and $E$ is a set of $n$ arcs, and let $\mathcal{K}$ be a set of $K$ commodities to be routed through the network represented by $G$.
Each commodity $k$, for $k \in K$, has a unique source $o^k$ and a unique sink $p^k$. We denote by $A \in \mathbb{R}^{(p-1)\times n}$ the matrix obtained by deleting an arbitrary row from the node-arc incidence matrix of $G$. (It is well known that this matrix is full row rank.) Each column of $A$ is related to an arc $e \in E$ and has only two nonzero $(1,-1)$ coefficients in those rows associated with (respectively) the origin and the destination nodes of $e$. We shall consider that the arcs of the network have a capacity $b^0$ $(\in \mathbb{R}^n)$ for all commodities. We denote by $b^k \in \mathbb{R}^{p-1}$, for $k \in K$, a vector of supplies/demands for commodity $k$ at the nodes of the network defined by:

$$b_i^k = \begin{cases} +r^k \text{ if } i = o^k, \\ -r^k \text{ if } i = p^k, \\ 0 \text{ elsewhere}, \end{cases} \tag{22}$$

where $r^k$ is the value of the flow of the $k^{th}$ commodity. Let us assume that an unitary cost vector $c = (c_e)_{e\in E}$ is given and that this cost does not depend on commodity. We shall distinguish two models of (LMNP): the node-arc model and the arc-path one.

## 3.2 Node-Arc Formulation

The node-arc formulation of (LMNP) can be expressed as follows:

$$\begin{cases} \min\limits_{x^1,\ldots,x^K} c^T \sum\limits_{k=1}^{K} x^k; & \\ (i)\ x^0 + \sum\limits_{k=1}^{K} x^k & = \ b^0, \\ (ii)\ Ax^k & = \ b^k,\ k = 1,\ldots,K, \\ (iii)\ x^k & \geq \ 0,\ k = 0,\ldots,K, \end{cases} \tag{23}$$

where

- $x^k$ is a $\mathbb{R}^n$-vector, $x_e^k$ is the flow carried by commodity $k$ on arc $e$, $(e \in E)$,

- $x^0$ is the residual capacity vector ($x^0 \in \mathrm{I\!R}^n_+$).

In the node-arc formulation, the decision variables are the flow of commodities on each arc. This formulation associates to each commodity $k$ flow conservation (23.*ii*) and non-negativity (23.*iii*) constraints. The equations (23.*i*) refer to the capacity constraints. The node-arc formulation of (LMNP) (23) has $\hat{p} = K \times (p-1) + n$ constraints and $\hat{n} = (K+1) \times n$ variables. For real networks such as the Paris district transmission network, the number of nodes can be larger than 100 and the number of arc larger than 300. If one assume that this network has 1000 commodities, the size of the problem can be larger than 100300 constraints and 300300 variables. So, the linear problem (23) is very large even for a small network.

We propose to specialize the method developed in the previous section to this problem. Let us note that the system (23) corresponds to the system (2). Indeed, in the case of the node-arc formulation of the linear multicommodity network-flow problem, we have:

- for all $i \in \{0, \ldots, K\}$: $n^i := n$ and $B^i := I(n)$.
- for all $i \in \{1, \ldots, K\}$: $A^i := A$, $p^i := p - 1$ and $c^i = c$.
- $p^0 := n$ and $c^0 := 0$.

We refer to this specialization of predictor-corrector algorithm to node-arc formulation of (LMNP) by the NAF algorithm.

## 3.3 Arc-path Formulation

The linear multicommodity network problem can be formulated in term of paths. The decision variables become the flow components on paths. We denote by $I^k$ the set of distinct elementary paths between the source node $o_k$ and the sink node $p_k$ in $G$. We designate by $n^k$ the cardinal of $I^k$ ($n^k = |I^k|$). An element $i$ of the set $I^k$ is characterized by a $\mathrm{I\!R}^n$-boolean vector $\{\pi_i^k\}$ satisfying:

$$(\pi_i^k)_e = \begin{cases} 1 \text{ if } e \text{ belongs to i }, \\ 0 \text{ otherwise.} \end{cases} \tag{24}$$

The variables $x^1, \ldots, x^K$, are now associated to elementary paths rather then to arcs. We denote by $x_i^k$ the component of the commodity $k$ carried by the path characterized by $\pi_i^k$. The sum over $i$ of the components $x_i^k$ is the value of the commodity $k$: $\sum\limits_{i \in I^k} x_i^k = r^k$. The total flow of the commodity $k$ is the $\mathrm{I\!R}^n$-vector $\sum\limits_{i \in I_k} x_i^k \pi_i^k \in \mathrm{I\!R}^n$. We recall that the $\mathrm{I\!R}^n$-vector $x^0$ is the residual capacity vector.

The arc-path formulation of the (LMNP) consist of the following program:

$$\left\{\begin{array}{lr}\displaystyle\operatorname*{Min}_{x^1,\ldots,x^K} c^T \sum_{k=1}^{K} \pi^k x^k; & \\ \displaystyle (i)\ x^0 + \sum_{k=1}^{K} \pi^k x^k = b^0, & (\lambda^0) \\ k \in \{1,\ldots,K\} \left\{\begin{array}{ll} (ii)\ \omega^k x^k = r^k, & (\lambda^k) \\ (iii)\ x_i^k \geq 0, i \in I^k, & (s_i^k) \end{array}\right. & \\ (iv)\ x^0 \geq 0, & (s^0) \end{array}\right. \tag{25}$$

where for all $k \in K$, $\omega^k$ is the $\mathbb{R}^{n^k}$ row-vector defined by $\omega_i^k = 1$, for $i = 1,\ldots,n^k$. We can recognize the same type of constraints as in the node-arc formulation. Indeed, the equations (25.*ii*) and (25.*i*) represent respectively the flow conservation constraints associated to each commodity and the capacity constraints.
The general formulation (2), is specialized for the (LMNP) arc-path formulation (25) by setting:

- for all $k \in \{1,\ldots,K\}$: $c^k := (\pi^k)^T c$, $(c^k \in \mathbb{R}^{n^k})$, $B^k = \pi^k$, $b^k = r^k$, $A^k = \omega^k$ and $p^k = 1$.
- $B^0 = I(n)$ and $p^0 = n$.

Then we can, as in the case of the node-arc formulation, adapt the algorithm described in the general framework of linear optimization with coupling constraints problems to the arc-path formulation of LMNP. We refer to this adaptation by APF. The arc-path formulation associates to each elementary path a variable. The number of variables may then increase exponentially with the size of the graph. This constitutes the main shortcoming for this formulation. However, in practice, an optimal solution is carried by a relatively small number of paths. Indeed the formulation (25 has only $n + K$ constraints. This implies that (if (25) has a solution) an optimal solution that has at most $n + K$ strictly nonzero variables exists. Then no more than $n + k$ different paths are needed to satisfy all the requested flow.
We propose to take advantage of this property by limiting the routing problem LMNP to a subset of elementary paths generated by an iterative process. Indeed, the path matrixes $\pi^k$ are only known implicitly. However, the form of such columns is known, and they can be generated as needed during the course of the algorithm- hence the name column generation [FF58, Jew58]. The scheme of column generation algorithm requires one to solve successive linear programs of smaller size. We use the algorithm PCLMNP for solving these programs. The algorithm APF does not take into account all the sets $I^k$. It starts with a subset $I_0^k$ and adds at each iteration $j$, no more than one path $\chi^{k,j}$ to the subset $I_j^k$. In other words APF introduce at each iteration $j$, no more than one column to the matrix $\pi^k$ . By reducing the number of the columns, we reduce also the number of variables.
We refer to the stage of paths (columns) generation as oracles. The Master Program consist

of the resolution of a routing problem using a limited number of paths. Indeed, at each iteration $j$ a commodity $k$ can be carried only by paths belonging to a subset $I_j^k$.
At each outer iteration j, algorithm APF solves the following master program:

$$\left\{\begin{array}{l} \displaystyle\operatorname*{Min}_{x^1,\dots,x^K} c^T \sum_{k=1}^{K} \pi^{k^j} x^k; \\ \displaystyle\text{(i) } x^0 + \sum_{k=1}^{K} \pi^{k^j} x^k = b^0, \\ k \in \{1,\dots,K\} \left\{\begin{array}{l} \text{(ii) } \omega^{k^j} x^k = r^k, \\ \text{(iii) } x_i^k \geq 0, i \in I^{j^k}, \end{array}\right. \\ \text{(iv) } x^0 \geq 0, \end{array}\right. \tag{26}$$

where

- $n^{k^j} = |I_j^k|$,
- $\omega^{k^j} \in \mathbb{R}^{n^{k^j}}$ and $\omega_i^{k^j} = 1,\ i = 1,\dots,n^{k^j}$,
- $x^k \in \mathbb{R}^{n^{k^j}}$,
- $\pi^{k^j}$ is the matrix whose columns are $\pi_i^k, i \in I_j^k$.

The optimality system associated to this problem is:

$$\left\{\begin{array}{lll} x^0 s^0 & = & 0, \\ \displaystyle x^0 + \sum_{k=1}^{K} \pi^{k^j} x^k & = & b^0, \\ \lambda^0 & = & s^0, \\ k \in \{1,\dots,K\} \left\{\begin{array}{lll} x^k s^k & = & 0, \\ \omega^{k^j} x^k & = & r^k, \\ (c^k + \lambda^0)^T \pi^{k^j} + \omega^{k^j} \lambda^k & = & s^k, \end{array}\right. & & \\ x^k, s^k \geq 0, \quad k = 0,\dots,K, & & \end{array}\right. \tag{27}$$

We denote by $(x^j, s^j, \lambda^j)$ the solution of problem (26) given by predictor-corrector algorithm. It satisfies the optimality system (27). We introduce $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ defined by :

$$\left\{\begin{array}{l} \tilde{x}^0 = x^{0^j}, \\ \tilde{s}^0 = s^{0^j}, \\ \tilde{\lambda}^0 = \lambda^{0^j}, \\ \text{for } k \in \{1,\dots,K\} \left\{\begin{array}{l} \tilde{x}_i^k = \left\{\begin{array}{l} x_i^k \text{ if } i \in I_j^k, \\ 0 \text{ else.} \end{array}\right. \\ \tilde{s}_i^k = (c^k + \lambda^{0^j})^T \pi_i^k + \lambda^{k^j},\ i \in I^k. \\ \tilde{\lambda}^k = \lambda^{k^j}. \end{array}\right. \end{array}\right. \tag{28}$$

For all commodity $k \in K$, we designate by $\chi^{k,j}$ a $\mathbb{R}^n$-boolean vector characterizing a shortest path between $o_k$ and $p_k$ calculated with $c^k + \lambda^{0j}$ as unitary cost vector. We have the following classical result:

**Lemma 2** *the following two statements are equivalent*

*(i) the point $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ defined by (28) is solution of (LMNP)*
*(ii) for all commodity $k$ in $\{1, \ldots, K\}$ we have:*

$$(c^k + \lambda^{0,j})^T \pi_j^k + \lambda^k \geq 0. \tag{29}$$

**Proof 2** *Let us first check that $(i)$ imply $(ii)$:*
*If $(\tilde{x}, \tilde{s}, \tilde{\lambda})$ is a solution of (25), then it satisfies the associated optimality system. We obtain this optimality system, by setting $I_j^k = I^k$ in (27).*
*The dual constraint $(c^k + \lambda^0)^T \pi_i^k + \lambda^k \geq 0$ is satisfied for all paths in $I^k$. In particular it remains true for $\chi^{k,j}$.*
*Let us check now that $(ii)$ leads to $(i)$:*
*For all paths in $I^k$ we have $(c^k + \lambda^0)^T \pi_i^k \geq (c^k + \lambda^0)^T \chi^{k,j}$.*
*If $(c^k + \lambda^{0j})^T \chi^{k,j} + \lambda^{kj} \geq 0$ then $(c^k + \lambda^{0j})^T \pi_i^k + \lambda^{kj} \geq 0$ for all $i \in I^k$.*
*Then $(\tilde{x}, \tilde{s}, \lambda)$ verifies the optimality system associated to (LMNP).*

**Management of the sets $I^k$**
At each outer iteration $j$ and for all commodity $k$ in $\{1, \ldots, K\}$, the algorithm APF computes a shortest path $\chi^{k,j}$. When the dual variable associated to this path is negative ($(c^k + \lambda^{0j})^T \chi^{k,j} + \lambda^{kj} < 0$), the algorithm adds the path $\chi^{k,j}$ to the set $I_j^k$ : $I_{j+1}^k = I_j^k + \chi^{k,j}$.
Now we are ready to state the (APF) algorithm.

**Algorithm APF**

- **Initialisation:** $j = 0$, for $k = 1, \ldots, K$:
  - Solve shortest path problems: $I_0^k$.
  - If $\sum_{k=1}^{K} {\pi^k}^0 r^k > b^0$ then add *artificial vector capacity* and use *Big M* method.
- **Master program:** Solve an (LMNP) limited to subsets $I_j^1, \ldots, I_j^K$ by predictor corrector algorithm PCLMNP.
  Deduce $(\tilde{x}, \tilde{s}, \tilde{\lambda})$, (28).
- **Oracle**: for $k = 1, \ldots, K$:
  Solve shortest path problem: Compute $\chi^{k,j}$ shortest path between $o_k$ and $p_k$ calculated with $c^k + \lambda^{0j}$.
- **Stop test**: $test = 0$,

- for $k = 1, \ldots, K$: if $(c^k + \lambda^{0j})^T \chi^{k,j} + \lambda^{kj} < 0$ then $I^k_{j+1} = I^k_j + \chi^{k,j}$ and $test = test + 1$.
- if $test > 0$ then $j = j + 1$, go to 2, else Stop.

## 4 Numerical Results

In this paper, the numerical tests were performed in MATLAB environment. For this reason, the computing time is not significant.

### 4.1 Randomly generated Problems

#### 4.1.1 Performance of algorithm NAF

We consider networks with $n = 60$ arcs and $p = 30$ nodes randomly generated. The number of commodities varies up to 1000. We denote by $K$, $i_a$ and $i_w$, respectively, the number of commodities, the average value of the number of iterations and the number of iterations in the worst case.
For each value of $K \in \{10, 50, 100, 500, 1000\}$, we generate randomly 10 networks.
The generation of a problem consists of the generation of node-arc incidence matrix of the graph $G = (V, E)$.
The starting point is $x^0 = s^0 = \mathbf{1}$ and $\mu_0 = 1$. The algorithm stops when $\mu < 10^{-10}$. The numerical results obtained by NAF are summarized in the table 1.

| K | $i_a$ | $i_w$ |
|---|---|---|
| 10 | 18.8 | 20 |
| 50 | 22.7 | 25 |
| 100 | 24.2 | 26 |
| 500 | 29 | 32 |
| 1000 | 35.20 | 39 |

Table 1: Performance of the algorithm NAF on Networks with 30 nodes and 60 arcs

#### 4.1.2 Comparison between NAF and APF performances

We generate randomly 5 LMNP. We test the performances of the algorithms for the case $p = 30$, $n = 70$, and $K = 25$.
We obtain the following table: where

- $t1$ is the computing time with (NAF).
- $t2$ is the computing time with (APF).

| K | t1 | t2 | i1 | i2 |
|---|---|---|---|---|
| pb1 | 376 | 80 | 42 | 15 |
| pb2 | 348 | 63 | 40 | 12 |
| pb3 | 324 | 80 | 37 | 15 |
| pb4 | 314 | 82 | 36 | 15 |
| pb5 | 305 | 82 | 37 | 15 |

Table 2: Comparison between NAF and APF performances

- $i1$ is the number of iterations with (NAF).
- $i2$ is the number of iterations with (APF).

We establish that the algorithm APF needs (in average) the one fourth of the computing time required by NAF.

## 4.2 Tests With Real Data

We report in this section some numerical results obtained by testing these algorithms with instances using real data from Paris district area transmission network. Since the information is classified, we are not allowed to give details about the data. We can only give the size of the problems in their standard form (see Table 3).
Our implementation of the algorithm APF uses the "Big M" method. The capacity of the generated paths may not be sufficient to meet all demands and one must add a supplement capacity with a very large cost (M) to assure the routing of the demands. Test NOE26 is particularly interesting. It is a highly degenerate problem as all the routing costs $c_i$ are equal to 1. Moreover, all the links are saturated in the optimal solution.
Test DRIF is encumbering because of the memory requirements. This explains why algorithm NAF was not able to treat this problem.

| p | n | K | $\hat{p}$ | $\hat{n}$ | name |
|---|---|---|---|---|---|
| 26 | 42 | 257 | 6981 | 34994 | `NOE26` |
| 119 | 302 | 900 | 108302 | 758102 | `DRIF` |

Table 3: Size of problems solved

Table 4 shows results obtained by NAF. In the table Iter is the number of iteration of the algorithm.

Unfortunately, we were not able to solve the problem DRIF because of the shortage of memory mentioned before.

| Iter | optimal cost | computing time (sec) | name |
|---|---|---|---|
| 17 | 9879.004 | 2472.883 | NOE26 |
| X | X | X | DRIF |

Table 4: Performance of the algorithm NAF

The results for APF are displayed in Table 5 where Outer Iter is the number of master programs solved and Inner Iter, the iteration number of predictor-corrector algorithm.

| Outer Iter | Inner Iter | optimal cost | computing time (sec) | name |
|---|---|---|---|---|
| 5 | 68 | 9879.022 | 1501.117 | NOE26 |
| 5 | 79 | 6040.601 | 51352.37 | DRIF |

Table 5: Performance of the algorithm APF

This numerical results show the superiority of APF.

## 5 Conclusion

We have developed routing algorithms for node-arc and arc-path formulation and we have obtained promising results. They both are suitable for a distributed implementation on a massively parallel computer. The algorithm NAF proposed in this paper can be improved because the aggregation of the commodities is possible in the case of node-arc formulation. Since the cost per unit of flow on a given link does not depend on the commodity, all the commodities witch have a common endpoint can be merged. Therefore the problem reduces to a number of commodities not larger than $p$ the number of nodes.

| p | n | K | $\hat{p}$ | $\hat{n}$ | name |
|---|---|---|---|---|---|
| 26 | 42 | 24 | 690 | 3306 | NOE26-merged |
| 119 | 302 | 63 | 7862 | 53348 | DRIF-merged |

Table 6: size of aggregated dates

DRIF problem can then be solved with NAF. We can also improve both of NAF and APF algorithms by solving the Newton Direction approximatively. The matrix $M$ is symmetric positive definite. Then one can use an iterative method to solve the reduced system as the conjugate gradient method.

## References

[AGMX96] E.D. Andersen, J. Gondzio, C. Mészáros, and X. Xu. Implementation of interior point methods for large scale linear programming. In ed. T. Terlaky, editor, *Interior point methods in mathematical programming*, pages 189–252. Kluwer academic publishers, Dordrecht, the Netherlands, 1996.

[AHKL80] A. Ali, R.V. Helgason, J.L. Kennington, and H. Lall. Computational comparison among three multicommodity network flow algorithms. *Operations Research*, 28:995–1000, 1980.

[AMO93] Ravindra K. Ahuja, Thomas L. Magnanti, and James B. Orlin. *Network flows*. Prentice Hall Inc., Englewood Cliffs, NJ, 1993. Theory, algorithms, and applications.

[Ass78] A. Assad. Multicommodity network flows-a survey. *Networks*, 8:37–91, 1978.

[BGLS97] J. Frédéric Bonnans, Jean Charles Gilbert, Claude Lemaréchal, and Claudia Sagastizábal. *Optimisation numérique*. Springer-Verlag, Berlin, 1997. Aspects théoriques et pratiques. [Theoretical and applied aspects].

[BPR00] J.F. Bonnans, C. Pola, and R. Rébaï. Perturbed path following interior point algorithms. *OMS*, 2000. To appear.

[Casar] J. Castro. A specialized interior point algorithm for multicommodity network flows. *SIAM J. Optimization*, to appear.

[CG90] In Chan Choi and Donald Goldfarb. Solving multicommodity network flow problems by an interior point method. In *Large-scale numerical optimization (Ithaca, NY, 1989)*, pages 58–69. SIAM, Philadelphia, PA, 1990.

[CL97] P. Chardaire and A. Lisser. Simplex and interior point specialized algorithms for solving non-oriented multicommodity flow problems. Technical report, CNET, 1997.

[CMR94] J. Chifflet, P. Mahey, and V. Reynier. Proximal decomposition for multicommodity flow problems with convex costs. *Telecommunication Systems*, 3:1–10, 1994.

[Dan61] G.B. Dantzig. The decomposition algorithm for linear programming. *Econometrica*, 29:767–778, 1961.

[DER89] I. S. Duff, A. M. Erisman, and J. K. Reid. *Direct methods for sparse matrices*. The Clarendon Press Oxford University Press, New York, second edition, 1989. Oxford Science Publications.

[DW60] G.B. Dantzig and P. Wolfe. Decomposition principle for linear programming. *Operations Research*, 8:101–111, 1960.

[FF58] L.R. Ford and D.R. Fulkerson. A suggested computation flor maximal multicommodity network flow. *Manag.Sci*, 5:97–101, 1958.

[FF62] L.R. Ford and D.R. Fulkerson. *Flows in Networks.* Princeton University Press, 1962.

[Fra97] A. Frangioni. *Dual ascent methods and multicommodity flows.* Dip. di informatica, Univ. di Pisa, 1997.

[GGSV97] J.-L. Goffin, J. Gondzio, R. Sarkissian, and J.-P. Vial. Solving nonlinear multicommodity flow problems by the analytic center cutting plane method. *Math. Programming*, 76(1, Ser. B):131–154, 1997. Interior point methods in theory and practice (Iowa City, IA, 1994).

[Gon92] C.C. Gonzaga. Path following methods for linear programming. *SIAM Review*, 34:167–227, 1992.

[GT97a] C.C. Gonzaga and R.A. Tapia. On the convergence of the Mizuno-Todd-Ye algorithm to the analytic center of the solution set. *SIAM J. Optim.*, 7(1):47–65, 1997.

[GT97b] C.C. Gonzaga and R.A. Tapia. On the quadratic convergence of the simplified Mizuno-Todd-Ye algorithm for linear programming. *SIAM J. Optim.*, 7(1):66–85, 1997.

[Jew58] W.S. Jewell. Optimal flow through networks. Technical report, Massachussets Institue of Technology, 1958.

[Ken78] J.F. Kennington. A survey of linear cost multicommodity network flows. *Oper.Res*, 2(26):209–236, 1978.

[KKR93] A.P. Kamath, N.K. Karmarkar, and K.G Ramakrishnan. Computational ans complexity results for an interior point algoritm on multicommodity flow problems. Technical Report TR-21/93, Dip. di Informatica, Univ. di Pisa, Italy, 1993. pp. 116-122.

[KMNY91] M. Kojima, N. Megiddo, T. Noma, and A. Yoshise. *A unified approach to interior point algorithms for linear complementarity problems.* Springer-Verlag, Berlin, 1991.

[KV96] S. Kapoor and P.M. Vaidya. Speeding up Karmarkar's algorithm for multicommodity flows. *Mathematical programming*, 73:111–127, 1996.

[LSV95a] A. Lisser, R. Sarkissian, and J.P. Vial. Optimal joint synthesis of base and reserve telecommunication networks. Technical report, CNET, Novembre 1995.

[LSV95b] A. Lisser, R. Sarkissian, and J.P. Vial. Survivability in transmission telecommunication networks. Technical report, CNET, April 1995.

[LY94] Zhi-Quan Luo and Yinyu Ye. A genuine quadratically convergent polynomial interior point algorithm for linear programming. In *Advances in optimization and approximation*, pages 235–246. Kluwer Acad. Publ., Dordrecht, 1994.

[Meh92] S. Mehrotra. On the implementation of a (primal-dual) interior point method. *SIAM J. Optimization*, 4(2):575–601, 1992.

[Meh93] S. Mehrotra. Quadratic convergence in a primal-dual method. *Mathematics of Operations Research*, 18:741–751, 1993.

[MTY93] S. Mizuno, M.J. Todd, and Y. Ye. On adaptive-step primal-dual interior-point algorithms for linear programming. *Mathematics of Operations Research*, 18:964–981, 1993.

[R0́0] R. Rébaï. *Optimisation de réseaux de télécommunications par des méthodes de points intérieurs.* PhD thesis, Université Paris Dauphine, Place du Maréchal de Lattre de Tassigny 75775 Paris cedex 16, Février 2000.

[RTV97] C. Roos, T. Terlaky, and J.-Ph. Vial. *Theory and algorithms for linear optimization.* John Wiley & Sons Ltd., Chichester, 1997. An interior point approach.

[Wri97] S.J. Wright. *Primal-dual interior-point methods.* SIAM, Philadelphia, PA, 1997.

[Ye92] Y. Ye. On the q-order of convergence of interior-point algorithms for linear programming. In Wu Fang, editor, *Proceedings Symposium on Applied Mathematics.* Institue of Applied Mathematics, Chinese Academy of Sciences, 1992.

[YGTZ93] Y. Ye, O. Güler, R.A. Tapia, and Y.Y. Zhang. A quadratically convergent $O(\sqrt{n}L)$ iteration algorithm for linear programming. *Mathematical Programming*, 59:151–162, 1993.

[Zen95] S. Zenios. A smooth penalty function algorithm for network-structured problems. *European Journal of Operational Research*, 83:220–236, 1995.