2007/91

■

A partitioning algorithm
for the network loading problem

Frédéric Babonneau and Jean-Philippe Vial

CORE DISCUSSION PAPER
2007/91

A partitioning algorithm for the network loading problem

Frédéric BABONNEAU [1] and Jean-Philippe VIAL[2]

November 2007

Abstract

This paper proposes a Benders-like partitioning algorithm to solve the network loading problem. The effort of computing integer solutions is entirely left to a pure integer programming solver while valid inequalities are generated by solving standard nonlinear multicommodity flow problems. The method is compared to alternative approaches proposed in the literature and appears to be efficient.

Keywords: network loading problem, Benders partitioning, ACCPM.

---

[1] Ordecsys, Switzerland.
[2] Ordecsys, Switzerland

This paper presents research results of the Belgian Program on Interuniversity Poles of Attraction initiated by the Belgian State, Prime Minister's Office, Science Policy Programming. The scientific responsibility is assumed by the authors.

1 Introduction

Let $\mathcal{G}(\mathcal{N}, \mathcal{A})$ be a directed graph where $\mathcal{N}$ is the set of nodes and $\mathcal{A}$ is the set of arcs. We denote $\mathcal{K}$ the set of OD pairs of demands. The network loading problem (NLP) consists of installing least cost capacities on the arcs that are sufficient to handle a network flow that meets the demands. As in [1] and [7], the capacities are assumed to be integer multiples of a base unit. NLP is the following mixed integer programming problem

$$\min_{y,x} \quad \sum_{a \in \mathcal{A}} r_a y_a \tag{1a}$$

$$\sum_{k \in \mathcal{K}} x_a^k \leq y_a, \qquad \forall a \in \mathcal{A}, \tag{1b}$$

$$N x^k = d_k \delta^k, \qquad \forall k \in \mathcal{K}, \tag{1c}$$

$$x_a^k \geq 0, \qquad \forall a \in \mathcal{A},\ \forall k \in \mathcal{K}, \tag{1d}$$

$$y_a \in \mathbb{N}, \qquad \forall a \in \mathcal{A}. \tag{1e}$$

Here, N is the network matrix; r_a is the cost of installing a unit capacity on arc a and the integer variable y_a represents the capacity on arc a; δ_k is the demand for the OD pair k; and d^k is vector with only two non-zeros components: 1 at the origin node and -1 at the destination node. The variable x^k is the flow for OD pair k on the arcs of the network. This problem is known to be strongly NP-hard on general graphs [7]. In general, the number of integer variables is very small compared to the number of flow fractional variables.

This problem has drawn a large attention in the literature [4]. The standard solution method for NLP is a cutting plane algorithm based either on the capacity formulation or on the flow formulation. The cutting plane algorithm solves the continuous relaxation of the NLP while heuristics are performed to find out an integer solution close to the fractional solution. These heuristics may be time consuming. A large variety of valid inequalities has been proposed to implement the cutting plane scheme. We can mention cut inequalities, 3-partition inequalities and arc residual capacity inequalities in [13]; flow cutset inequalities in [8, 9]; tight metric inequalities in [1]; partition inequalities and total capacity inequalities in [7].

Our solution method is a Benders partitioning scheme. The master program is a pure integer programming problem in the space of capacities. The subproblem is a continuous optimization problem; it tests whether the integer capacities generated by the master program is sufficient to support flows that meet the demands. It also computes supporting hyperplanes to the set of feasible fractional capacities. The subproblem makes it possible to construct a polyhedral relaxation of the set of feasible capacities. The master program looks for interesting integer solutions within that polyhedral relaxation.

The effort of computing integer solutions is entirely left to a pure integer programming solver (CPLEX in our case). The proposed solution method does not need heuristic to compute an integer solution and does not attempt to incorporate refinements of the cuts to get closer the convex hull of feasible integer capacities. The essential difference with the classical Benders partitioning scheme is that the master program does not search for a least cost integer solution within the relaxation. Rather, it looks for an improving integer solution that is nearest to the best feasible integer solution generated so far. The subproblem is a standard nonlinear multicommodidity flow problem. We solve it with a Matlab version of OBOE [14], a solver for convex nondifferentiable optimization based on ACCPM (Analytic Center Cutting Plane Method).

The main advantage of our solution method is its simplicity and its efficiency on some problems. We improve the best upper bound found in the literature on some instances.

The method has two main drawbacks. First, it does not generate lower bound during the process. Only one lower bound is computed in the initialization phase that is the rounded fractional capacities from continuous relaxation of NLP. The second drawback is its reliance on a integer solver (commercial in our case).

2 Capacity formulation

For the sake of simpler notation, let us consider the equivalent formulation

$$\min_{y,x}\{r^T y \mid Ax \leq y, x \in \mathcal{X}, y \in \mathbb{N}^n\}, \tag{2}$$

where $\mathcal{X}$ is the set of feasible flows and the matrix A collects the flows on the individual arcs. Problem (2) involves a few integer variables and very many continuous variables. It is possible to give an alternative formulation in the y variables only. Let us define the set of feasible fractional capacities as

$$\mathcal{Y} = \{y \in \mathbb{R}^n_+ \mid \exists x \in \mathcal{X} \text{ such that } Ax \leq y\}.$$

This set is the continuous relaxation of the feasible set of (2). Note that $\mathcal{Y}$ is the projection in the y space of a polyhedral set in the (x, y) space. It is thus a polyhedral set and it can be described by a finite set of inequalities that we shall denote $By \leq b$ thereafter. Thus

$$\mathcal{Y} = \{y \in \mathbb{R}^n_+ \mid By \leq c\}. \tag{3}$$

In the literature these inequalities are referred to as Metric Inequalities. With these notations problem (2) is equivalent to the following problem

$$\min_{y}\{r^T y \mid y \in \mathcal{Y} \cap \mathbb{N}^n\}, \tag{4}$$

which is known as the *capacity formulation* [7].

It is not possible to formulate explicitly the set of inequalities that define $\mathcal{Y}$ in (3). However it is relatively easy to construct polyhedral relaxations of this set. Suppose that $\bar{\mathcal{Y}} = \{y \in \mathbb{R}^n_+ \mid \bar{B}y \leq \bar{c}\}$ is one such relaxation, we have $\mathcal{Y} \subset \bar{\mathcal{Y}}$. It follows that the problem

$$\min_{y}\{r^T y \mid y \in \bar{\mathcal{Y}} \cap \mathbb{N}^n\}$$

is a relaxation of (4). If y^* solves it and $y^* \in \mathcal{Y}$, then y^* is optimal to (2). In view of this short discussion we propose a conceptual Benders-like partitioning algorithm. We describe the basic iteration.

- **Initial data for the basic iteration**
  - `A set` $\bar{B}^k y \leq \bar{c}^k$ `of inequalities defining the relaxation` $\bar{\mathcal{Y}}^k$ `of` $\mathcal{Y}$.
  - `A feasible point` $y^k \in \mathcal{Y} \cap \mathbb{N}^n$.
- **Master iteration**
  - `Find` $\hat{y} \in \bar{\mathcal{Y}}^k \cap \mathbb{N}^n$ `such that` $r^T \hat{y} < r^T y^k$. `If there is no such` $\hat{y}$, `terminate;` y^k `is an optimal solution.`
- **Subproblem iteration** (Feasibility test)
  - `If` $\hat{y} \in \mathcal{Y}$, `update` $y^{k+1} = \hat{y}$ `and` $\bar{\mathcal{Y}}^{k+1} = \bar{\mathcal{Y}}^k$.

– `If` $\hat{y} \notin \mathcal{Y}$`, find a vector` $b^{k+1} \in \mathbb{R}^n$ `and` $c^{k+1} \in \mathbb{R}$ `such that` $(b^{k+1})^T \hat{y} > c^{k+1}$ `and` $(b^{k+1})^T y \leq c^{k+1}$ `for all` $y \in \mathcal{Y}$`.`
`Update` $\bar{\mathcal{Y}}^{k+1} = \bar{\mathcal{Y}}^k \cap \{y \mid (b^{k+1})^T y \leq c^{k+1}\}$ `and` $y^{k+1} = y^k$`.`

To make the algorithm operational, we have to explicit the computation to be performed in the master iteration and in the subproblem iteration. In the standard Benders decomposition scheme, the master iteration selects the best point in the relaxation

$$\hat{y} = \arg\min\{r^T y \mid y \in \bar{\mathcal{Y}}^k \cap \mathbb{N}^n\}.$$

This strategy is inefficient because the chosen point usually turns out to be very far from the feasible set $\mathcal{Y}$. Moreover, computing an optimal point of a the pure integer programming relaxation is more than often very demanding. We shall present an alternative in the next section.

The computation in the subproblem iteration consists in finding a hyperplane that separates the candidate point $\hat{y}$ from the convex set $\mathcal{Y}$. This can be done by solving some kind of convex programming problem. Of course, one would like the separation as deep as possible. We shall discuss several strategies to achieve this goal.

3 Benders master problem iteration

We now make precise the second step of the conceptual algorithm. As pointed out, the classical Benders decomposition scheme selects the best integer point in the relaxed capacity feasible set. This strategy is not appropriate. Assume for instance that the current relaxation includes a single inequality —a situation that occurs at the first iteration—. As we shall see, this inequality $b^1 y \leq c^1$ has the property—in our problem of interest—that $b^1 \leq 0$ and $c^1 < 0$. Benders decomposition selects

$$y^2 = \arg\min\{r^T y \mid y \in \mathcal{Y}^1 \cap \mathbb{N}^n\}.$$

This point will be close to one of the extreme points of the simplex $(b^1)^T y = c^1$ and is likely to be almost irrelevant for the capacity problem. The second objection to Benders scheme is that finding the best integer point is a difficult task, even for efficient integer programming solvers. In view of these potential shortcomings, we propose a new strategy:

> Given a current feasible capacity $\bar{y} \in \mathcal{Y} \cap \mathbb{N}^n$, find $y \in \mathbb{N}^n$ which is closest to $\bar{y}$ (relatively to some well-chosen norm) and such that $r^T y < r^T \bar{y}$.

This problem turns out to be easier. Moreover, it produces points that are close to the feasible capacity $\bar{y}$ and, as such, that are likely to be informative.

To implement this strategy, we need an initial feasible point for (1). We propose a simple heuristic: solve the LP-relaxation of (1) and round up its fractional optimal solution. The resulting capacity is feasible and is used as the starting point for the heuristic.

Let $\bar{y}$ be a feasible capacity installation, the main iteration consists in finding a new feasible integer point close to $\bar{y}$ that improves the objective function value $r^T \bar{y}$. We denote by $\eta^+ \in \mathbb{N}^n$ and $\eta^- \in \mathbb{N}^n$ the capacity increase and capacity decrease for $\bar{y}$, respectively. Then $y - \bar{y} = \eta^+ - \eta^-$ and

$$\sum_{a \in \mathcal{A}} (\eta_a^+ + \eta_a^-) = \sum_{a \in \mathcal{A}} |y_a - \bar{y}_a|,$$

when $\eta_a^+\eta_a^- = 0$ for all a. With this notation, the inequality $r^T y < r^T \bar{y}$ is equivalent to $r^T(\eta^+ - \eta^-) < 0$. Finally, we replace the condition $\bar{y} + \eta^+ - \eta^- \in \bar{\mathcal{Y}}$ by

$$\bar{B}(\eta^+ - \eta^-) \leq \bar{c} - \bar{B}\bar{y} \text{ and } \eta^- \leq \bar{y}.$$

The main iteration solves

$$\min_{\eta^+,\eta^-} \quad \sum_{a\in\mathcal{A}} (\eta_a^+ + \eta_a^-) \tag{5a}$$
$$r^T(\eta^+ - \eta^-) < 0, \tag{5b}$$
$$\bar{B}(\eta^+ - \eta^-) \leq \bar{c} - \bar{B}\bar{y}, \tag{5c}$$
$$\bar{y} \geq \eta^-, \tag{5d}$$
$$\eta^+ \in \mathbb{N}^n, \quad \eta^- \in \mathbb{N}^n. \tag{5e}$$

If the pair $(\hat{\eta}^+, \hat{\eta}^-)$ solves (5), then the candidate point is $\hat{y} = \bar{y} + \hat{\eta}^+ - \hat{\eta}^-$.

4 Metric inequalities for the subproblem

Let $\hat{y} \in \mathbb{N}^n$ be an integer capacity vector, possibly infeasible. One wants to test

$$\hat{y} \in \mathcal{Y} = \{y \mid \exists x \in \mathcal{X} \text{ such that } Ax \leq y\}.$$

To this end, we introduce a convex function $f : \mathbb{R}_+^n \to \mathbb{R}$ with the property that $f(0) = 0$ and $f(\beta) > 0$ if $\beta \neq 0$. We solve the problem

$$\delta(\hat{y}, \mathcal{Y}) = \min_{x,\beta\geq 0} \{f(\beta) \mid Ax \leq \hat{y} + \beta, x \in \mathcal{X}\}. \tag{6}$$

The function $\delta(\hat{y}, \mathcal{Y})$ is non negative and takes the value 0 iff $y \in \mathcal{Y}$.

We resort to the Lagrangian duality to compute $\delta(\hat{y}, \mathcal{Y})$. By duality, the min value is equal to the optimum of the dual problem

$$\delta(\hat{y}, \mathcal{Y}) = -\hat{y}^T u + \max_{u\geq 0} \{g_1(u) + g_2(u)\},$$

with

$$g_1(u) = -\min_{\beta\geq 0} (f(\beta) - \beta^T u), \tag{7}$$

and

$$g_2(u) = \min\{(A^T u)^T x \mid x \in \mathcal{X}\}. \tag{8}$$

The function $\delta(\hat{y}, \mathcal{Y})$ is convex, since it is the maximum of a family of linear forms in $\hat{y}$. If u^* is such that

$$\delta(\hat{y}, \mathcal{Y}) = -\hat{y}^T u^* + g_1(u^*) + g_2(u^*),$$

we have the following separating dual-based hyperplane for $\hat{y}$ such that $\delta(\hat{y}, \mathcal{Y}) > 0$

$${u^*}^T y \geq g_1(u^*) + g_2(u^*), \ \forall y \in \mathcal{Y}. \tag{9}$$

The g_2 function

Problem (8) is a linear programming problem. More precisely, in the multicommodity case, the problem boils down to independent shortest paths problems, one for each commodity. Very efficient techniques are used to solve this problem [10].

The g_1 function

The range of admissible functions $f(\beta)$ is large, but a norm is particularly well-suited $f(\beta) = ||\beta||_p$, $p \geq 1$. We give in Table 1, the functions $f(\beta)$ we use in the experiments. Let us explicit the computation in Table 1 for $f(\beta) = ||\beta||_\infty$. With this norm, problem

p	$f(\beta)$	$g_1(u)$ $(u \geq 0)$
1	$\|\|\beta\|\|_1$	$u \leq e$
∞	$\|\|\beta\|\|_\infty$	$e^T u \leq 1$
$1 < p < \infty$	$\|\|\beta\|\|_p^p$	$-(1-\frac{1}{p})\sum_i (u_i)^{\frac{p}{p-1}}$

Table 1: Some functions $f(\beta)$

(6) is equivalent to the scalar optimization

$$\delta(\hat{y}, \mathcal{Y}) = \min\{f(\beta) \mid Ax \leq \hat{y} + \gamma e,\ \beta = \gamma e,\ \gamma \in \mathbb{R}_+,\ x \in \mathcal{X}\}.$$

Indeed, the scalar γ can be interpreted as an upper bound on the components of β in (6). Thus

$$g_1(u) = -\min_{\gamma \geq 0} \gamma(1 - (e^T u)).$$

The optimum is either 0 if $e^T u \leq 1$ and $-\infty$ otherwise. Therefore, we set $g_1(u) = 0$, and add the constraint $e^T u \leq 1$. Actually, the equality holds, because for $\hat{y} \notin \mathcal{Y}$, then $\gamma > 0$ at the optimum and the complementarity condition $\gamma(e^T u - 1) = 0$ implies $e^T u = 1$.

The case $f(\beta) = ||\beta||_1$, with $\beta \geq 0$, is equivalent to $f(\beta) = e^T \beta$, a function that is commonly used to generate metric inequalities. In that case $g_1(u)$ is easily computed to be either 0 if $u \leq 0$ or $+\infty$ otherwise. Finally, the computations in the case $1 < p < +\infty$ are a little more tedious, but yet straightforward.

Suboptimal cutting planes

The separating hyperplane (9) is defined with respect to the optimal solution u^* of the Lagrangian dual problem. This solution cannot be computed exactly, but it is still possible to compute a separating hyperplane with an approximate solution. Suppose we observe

$$-\hat{y}^T \hat{u} + g_1(\hat{u}) + g_2(\hat{u}) > 0, \qquad \text{for some } \hat{u} \neq u^*. \tag{10}$$

Since $-\hat{y}^T u^* + g_1(u^*) + g_2(u^*) \geq -\hat{y}^T \hat{u} + g_1(\hat{u}) + g_2(\hat{u})$, we can safely state that

$$-y^T \hat{u} + g_1(\hat{u}) + g_2(\hat{u}) \leq 0$$

separates $\hat{y}$ and $\mathcal{Y}$.

The cutting plane with $1 < p < \infty$

The choice of this norm yields an alternative primal-based cutting plane that is unique, but requires high precision in the minimization of g_2. To see this, let $y^* = \hat{y} + \beta^*$ with

$$\beta^* = \arg \min_{\beta \in \mathbb{R}^n_+, x} \{f(\beta) \mid Ax \leq \hat{y} + \beta, x \in \mathcal{X}\}. \tag{11}$$

The point y^* is unique, because f is strictly convex for $1 < p < \infty$. Consider the level set

$$L = \{y \mid f(y - \hat{y}) \leq f(y^* - \hat{y})\}.$$

Clearly $\text{int}L \cap \mathcal{Y} = \emptyset$. Because f is smooth, the tangent plane at y^* is well-defined and unique. The separation hyperplane is thus

$$\left(f'(y^* - \hat{y})\right)^T (y - y^*) \geq 0,\ \forall y \in \mathcal{Y}.$$

In the case of the 2-norm, we obtain the simple inequality

$$(y^* - \hat{y})^T (y - y^*) \geq 0,\ \forall y \in \mathcal{Y}. \tag{12}$$

Recall that the normal to the supporting hyperplane also belongs to the negative of the normal cone to $\mathcal{Y}$ at y^*. This normal cone may include many other elements (the boundary of $\mathcal{Y}$ is not smooth) and the elements in that cone are all valid candidates to define a supporting plane to $\mathcal{Y}$. The minimum norm approach with $1 < p < \infty$ selects a single element in that set, a choice that is likely to produce a more efficient cutting plane in the master iteration. The drawback of this approach is that y^* must be computed with high accuracy to yield a valid separation.

The above argument falls apart for the extreme cases $p = 1$ and $p = \infty$ because the corresponding norms are not strictly convex (the minimizing point is not unique) and not smooth.

5 The full algorithm

The initialization phase and the main steps that compose the algorithm are described below.

1. **Initialization :**
   (a) To select $y^0 \in \mathcal{Y} \cap \mathbb{N}^n$, we solve the LP-relaxation of (1) and round up its fractional optimal solution.
   (b) Select a norm ℓ_p used in step 3 to generate the metric inequalities.
   (c) Initialize the relaxation of the feasible capacity set $\bar{\mathcal{Y}}^0 = \mathbb{R}^n_+$.
   (d) Fix a CPU time limit T.
   (e) $I_0 = 0$.
2. **Benders master iteration :**
   (a) If CPU time > T, terminate; $r^T y^k$ is a valid upper bound for (1).
   (b) Find $\hat{y} \in \bar{\mathcal{Y}}^k \cap \mathbb{N}^n$ by solving (5) with the pure integer programming solver.
   (c) If there is no such $\hat{y}$,terminate; y^k is an optimal solution.
3. **Feasibity check and metric inequalities :** ACCPM solves (6) with norm ℓ_p.
   (a) If $f(\beta) = 0$, $\hat{y}$ is feasible, i.e., $y \in \mathcal{Y}$. Update $y^{k+1} = \hat{y}$ and remove all metric inequalities $\bar{\mathcal{Y}}^{k+1} = \mathbb{R}^n_+$. Set I_{k+1} = 0.
   (b) If $f(\beta) > 0$, $\hat{y} \notin \mathcal{Y}$. Compute a metric inequality $b^T y \leq c$ using (9) or (12). Update $\bar{\mathcal{Y}}^k = \bar{\mathcal{Y}}^k \cap \{y \mid b^T y \leq c\}$ and $I_k = I_k + 1$.
   (c) Go to 2.

The index k corresponds to the number of times the solution method improves the upper bound in 3(a). The counter I_k gives the number of metric inequalities (and/or the number of Master iterations) until the k-th improvement of the upper bound. The total number of metric inequalities generated is denoted $MI = \sum_{\kappa=0}^{k} I_\kappa$ in the tables below.

Remark 1 *Note that the update $\bar{\mathcal{Y}}^{k+1} = \mathbb{R}^n_+$ in step 3(a) removes all previously generated metric inequalities, even though those inequalities are still valid. This choice is motivated by our empirical observation that the pure integer programming solver in the master iterations dangerously slows down when the number of metric inequalities (constraints) becomes large.*

6 Numerical experiments

The main goal of our empirical study is to test the efficiency of our partitioning algorithm using different metric inequalities. We use published results to benchmark the proposed algorithm.

6.1 Test problems

Our test bed is made of the two sets of Asymmetric Norwegian instances, named Sun.tr and Sun.dense, that have been used in [1, 7]. Each network has 27 nodes and 102 directed arcs. The Sun.tr instances have 67 OD-demand pairs with magnitude in the interval $[0.1, 0.2]$. All Sun.tr instances have been solved optimally in [1]. Because of greater congestion, the Sun.dense instances are considered to be more difficult; only bounds on the optimal solution are known. Each Sun.dense instance has 702 OD-demand pairs with magnitude in the interval $[1, 2]$.

The tests were performed on a PC (Pentium IV, 2.8 GHz, 2 Gb of RAM) under Linux operating system. The metric inequalities have been generated using the solution method based on ACCPM and as described in [3]. We used CPLEX 8.1 to solve the integer programming problem in the master iterations. The overall shell is written in Matlab as well as our version of ACCPM.

6.2 Algorithmic settings

The settings of CPLEX 8.1 are the default ones. To solve (6), we essentially used the default settings with ACCPM, but we varied the optimality tolerance level depending on the norm used in $f(\beta)$. We know that dual-based valid inequalities (9) can be generated at suboptimal points, more precisely as soon as (10) is satisfied, a situation that may occur with a non-negligible relative optimality gap. Of course, if the dual values gets closer to the optimum, the deeper is inequality (9). Nevertheless, the required level on the relative optimality gap in ACCPM is not an issue. In contrast, we need a good quality of the primal solution y^* to generate the primal-based inequality (12), and thus a very small relative duality gap in ACCPM. Table 2 gives our choice for the optimality tolerance parameters.

6.3 Synthesis of results

To get at a glance the comparative performance between our algorithm and the algorithms in [1] and [7], we put in Table 3 the results with [1] and [7] and a synthesis of the results

inequality	dual-based (9)			primal-based (12)
norm	ℓ_1	ℓ_2	ℓ_∞	ℓ_2
Optimality gap	10^{-3}	10^{-6}	10^{-5}	10^{-7}

Table 2: ACCPM settings for metric inequality

obtained with our method. A detailed account of the results with our algorithm is given in Section 6.4. The figures in the table are upper bounds on the optimal objective function value. The first column, denoted *UB in [1]*, in the best upper bound computed in [1] by solving the capacity formulation of (1). The last two ones are reported from [7]. UB1 and UB2 are the best upper bounds computed by solving the capacity formulation and the multicommodity formulation of (1), respectively. For our Benders-like algorithm, we report the best upper bound from Tables 5, 6, 7 and 8. In parentheses we give the norm used to generate metric inequalities and the CPU time to reach the upper bound.

	Literature			Partitioning
Problem ID	UB in [1]	UB1 in [7]	UB2 in [7]	Best UB (Norm, CPU)
Sun.tr1	**2962.42***	3027.3	2976.3	2990.76 (ℓ_2*, 1h)
Sun.tr2	**2976.24***	3013.6	2978.2	3007.46 (ℓ_2*, 3h)
Sun.tr3	**3242.78***	3309.9	3256.8	3262.22 (ℓ_2*, 1h)
Sun.tr4	**2978.90***	2979.4	**2978.9***	3026.30 (ℓ_2*, 4h)
Sun.tr5	**2585.00***	2633.4	2592.4	2591.18 (ℓ_∞, 3h)
Sun.tr6	**3196.96***	3282.9	3246.6	3238.51 (ℓ_2, 3h)
Sun.dense1	30265.1	30032	29804	**29781.75** (ℓ_∞, 3h)
Sun.dense2	30219.9	30211	29835	**29773.91** (ℓ_2*, 4h)
Sun.dense3	99329.7	100748	98829	**98760.62** (ℓ_∞, 4h)
Sun.dense4	99092.4	99839	98556	**98554.18** (ℓ_1, 2h)
Sun.dense5	59847.5	60178	59337	**59317.42** (ℓ_∞, 3h)
Sun.dense6	59667.5	59696	59130	**59121.20** (ℓ_2*, 4h)

* ℓ_2 norm with $(y - y^*)^T(\bar{y} - y^*) \leq 0$.

Table 3: Solutions in the literature

To make those results more readable, we display in Table 4 the same results, but in terms of the relative gap of the current solution with respect to the best solution achieved by the 4 algorithms. Surprisingly enough, our algorithm performs better on the more difficult problems Sun.dense than on the easier Sun.tr. On the former, our algorithm does better than the three algorithms in [1] and [7]. On the latter, our algorithm does rather better than UB1 [7], rather worse than UB2 [7] and worse than [1].

6.4 Detailed results

In this section we give a more detailed account on the behavior of our algorithm when the metric inequalities are derived from different norms. For each norm, we run our algorithm four hours on all the instances. The results are reported in Table 5 for the ℓ_1 norm, in Table 6 for the ℓ_2 norm, in Table 7 for the ℓ_∞ norm, and in Table 8 for the ℓ_2 norm using the metric inequality (12). In each case, we report the results after 1 hour, 2 hours and 4 hours. For all results, the tables give the upper bound, denoted *UB*, the number of times the algorithm improves the objective function, denoted *It*, and the number of generated metric inequalities, denoted *MI*.

	Literature			Benders
Problem ID	UB in [1]	UB1 in [7]	UB2 in [7]	Best UB
Sun.tr1	0	2.190	0.469	0.957
Sun.tr2	0	1.255	0.066	1.049
Sun.tr3	0	2.070	0.432	0.600
Sun.tr4	0	0.017	0	1.591
Sun.tr5	0	1.872	0.286	0.239
Sun.tr6	0	2.688	1.553	1.300
Average	0	1.6821	0.4676	0.9558
Sun.dense1	1.623	0.840	0.075	0
Sun.dense2	1.498	1.468	0.205	0
Sun.dense3	0.576	2.012	0.069	0
Sun.dense4	0.546	1.304	0.002	0
Sun.dense5	0.894	1.451	0.033	0
Sun.dense6	0.924	0.972	0.015	0
Average	1.010	1.341	0.067	0

Table 4: Relative gap in % with respect to the best solution

In all the tables, we use bold face characters to emphasize the production of a value that improves the best result of the literature. For instance, in Table 5, the algorithm produces a better value for Sun.dense2 after one hour of computing time. This result is subsequently improved after 2 hours, but no better value is obtained in the next two hours. More precisely, between hour 2 and hour 4 the integer programming solver received $2006 - 1825 = 181$ metric inequalities but could not produce a better integer solution.

	1 hour			2 hours			4 hours		
Problem ID	UB	It.	MI	UB	It.	MI	UB	It.	MI
Sun.tr1	3006.55	27	1258	3001.11	28	1539	-	-	1631
Sun.tr2	3016.31	43	1444	-	-	1529	3013.87	44	1818
Sun.tr3	3283.22	29	1385	3278.07	31	1771	3268.13	36	2437
Sun.tr4	3065.14	25	1111	3062.03	26	1350	3056.73	28	1613
Sun.tr5	2613.36	35	1428	-	-	1508	-	-	1561
Sun.tr6	3256.42	49	1669	3246.67	54	2433	-	-	2648
Sun.dense1	**29802.28**	37	982	**29784.96**	40	1471	-	-	1546
Sun.dense2	**29828.17**	51	1151	**29796.53**	59	1825	-	-	2006
Sun.dense3	98861.68	52	795	**98805.01**	66	1476	**98794.72**	69	2079
Sun.dense4	98583.20	36	786	**98554.18**	44	1371	-	-	1546
Sun.dense5	59380.86	40	934	**59335.05**	51	1624	**59318.54**	57	2393
Sun.dense6	59174.55	35	899	59163.59	40	1392	59163.47	41	1974

Table 5: Using metric inequalities from ℓ_1 norm

The next table, Table 9, summarizes the behavior of our algorithm with the different norms to generate the metric inequalities. It appears that the L2-norm* (with the separating hyperplane (12)) is more efficient. However, we can observe that on the Sun.tr instances, the difference among the four approaches is less than 0.8 % in average while the same figure drops to 0.06 % for the Sun.dense instances. Our algorithm with the L2-norm* is better than UB2 on all Sun.dense instances, except # 4 by a short margin.

The last two tables, Table 10 and (11), give the average time to generate one metric inequality for each norm and for each time interval and the proportion of time spent in computing metric inequalities. We observe that the average time is relatively constant during the processing and is not affected by the proximity to optimality. We also observe

Problem ID	1 hour UB	It.	MI	2 hours UB	It.	MI	4 hours UB	It.	MI
Sun.tr1	3003.02	22	1433	2997.31	23	1834	-	-	1982
Sun.tr2	3033.61	26	1651	3030.08	28	2249	-	-	2474
Sun.tr3	3281.47	27	1619	3274.28	29	2155	3269.13	32	3076
Sun.tr4	3075.10	22	1302	3061.25	25	1937	-	-	2135
Sun.tr5	2668.78	26	1341	2658.46	30	2210	2642.57	34	3483
Sun.tr6	3251.55	26	1616	3249.51	27	2008	3238.51	29	2537
Sun.dense1	29815.03	30	1274	**29797.28**	33	2044	**29790.23**	34	2540
Sun.dense2	29850.49	34	1401	29839.11	38	2138	**29824.32**	42	2861
Sun.dense3	98841.62	30	945	**98813.41**	36	1693	**98780.54**	40	2570
Sun.dense4	98624.51	32	1076	98586.87	44	2035	98564.52	50	3206
Sun.dense5	59406.41	38	1335	59364.37	46	2312	59341.45	51	3120
Sun.dense6	59197.15	35	1324	59166.40	42	2280	59164.92	44	3019

Table 6: Using metric inequalities from ℓ_2 norm

Problem ID	1 hour UB	It.	MI	2 hours UB	It.	MI	4 hours UB	It.	MI
Sun.tr1	2994.43	25	958	-	-	973	-	-	992
Sun.tr2	3016.390000	34	1330	-	-	1353	-	-	1367
Sun.tr3	3264.99	23	779	-	-	786	-	-	793
Sun.tr4	3072.49	23	856	3061.25	26	1064	-	-	1094
Sun.tr5	2652.68	23	609	2595.53	38	1756	2591.18	41	2127
Sun.tr6	3261.34	27	857	-	-	882	-	-	911
Sun.dense1	**29789.31**	44	1058	**29784.43**	45	1186	**29781.75**	47	1325
Sun.dense2	**29800.54**	50	1428	**29796.16**	54	1829	**29794.03**	57	2267
Sun.dense3	**98779.06**	50	851	**98770.23**	56	1304	**98760.62**	60	1769
Sun.dense4	98597.38	41	963	98561.37	53	1591	-	-	1629
Sun.dense5	59358.54	46	1142	**59325.19**	59	1991	**59317.42**	63	2369
Sun.dense6	59152.22	48	1127	-	-	1211	-	-	1241

Table 7: Using metric inequalities from ℓ_∞ norm

Problem ID	1 hour UB	It.	MI	2 hours UB	It.	MI	4 hours UB	It.	MI
Sun.tr1	2990.76	19	596	-	-	676	-	-	716
Sun.tr2	3008.55	29	918	-	-	952	3007.46	30	1243
Sun.tr3	3262.22	30	835	-	-	872	-	-	907
Sun.tr4	3050.79	34	1288	-	-	1388	3026.30	42	2489
Sun.tr5	2595.72	38	1280	-	-	1339	-	-	1370
Sun.tr6	3266.05	42	1431	-	-	1611	-	-	1694
Sun.dense1	29803.36	53	1284	**29787.89**	62	1997	**29783.53**	63	2311
Sun.dense2	29815.76	53	1284	**29785.31**	63	2138	**29773.91**	68	2963
Sun.dense3	98836.10	47	1009	**98824.83**	56	1788	**98816.56**	61	2624
Sun.dense4	98600.84	54	1001	98579.24	62	1587	98562.40	67	2418
Sun.dense5	59348.62	49	1002	59337.74	52	1416	**59333.48**	55	2058
Sun.dense6	59174.80	44	1033	59153.61	51	1643	**59121.20**	62	2841

Table 8: Metric inequalities $(y - y^*)^T(\bar{y} - y^*) \leq 0$ with the ℓ_2 norm

	L1-norm	L2-norm	L_∞-norm	L2-norm*
Sun.tr1	0.346	0.219	0.123	0
Sun.tr2	0.213	0.752	0.297	0
Sun.tr3	0.181	0.212	0.085	0
Sun.tr4	1.006	1.155	1.155	0
Sun.tr5	0.856	1.983	0	0.175
Sun.tr6	0.252	0	0.705	0.850
Average	0.476	0.720	0.394	0.171
Sun.dense1	0.011	0.028	0	0.006
Sun.dense2	0.076	0.169	0.068	0
Sun.dense3	0.035	0.020	0	0.057
Sun.dense4	0	0.010	0.007	0.008
Sun.dense5	0.002	0.041	0	0.027
Sun.dense6	0.071	0.074	0.052	0
Average	0.032	0.057	0.021	0.016

* ℓ_2 norm with $(y - y^*)^T(\bar{y} - y^*) \leq 0$.

Table 9: Optimality gap in percent with respect to the best result

that as the algorithm gets closer to the solution, the computing of metric inequalities takes far less time. In contrast, finding an improving integer solution is more and more difficult; the integer programming solver dramatically slows down the whole process.

Average	L1-norm	L2-norm	L_∞-norm	L2-norm*
	Sun.tr instances			
h0 to h1	1.43	1.2	0.74	1.54
h1 to h2	1.46	1.26	1.12	1.54
h2 to h4	1.48	1.24	0.75	1.54
	Sun.dense instances			
h0 to h1	3.71	2.84	3.03	3.00
h1 to h2	3.97	3.1	3.19	3.13
h2 to h4	4.08	3.02	3.23	3.23

* ℓ_2 norm with $(y - y^*)^T(\bar{y} - y^*) \leq 0$.

Table 10: CPU time in seconds per metric inequality with ACCPM

7 Conclusion

We have proposed a new algorithm to solve the network loading problem. At the upper level, the method works on the space of integer capacity variables. Contrary to previous approaches, it does not exploit the original mixed integer programming formulation of the problem. Rather, it uses metric inequalities to approximate the set of feasible capacities. The task of generating integer solutions in the space of the capacity variables is left to the integer programming solver. This solver uses the machinery of valid inequalities of various types and branch and bound schemes, but this is not visible to the user. At a lower algorithmic level, the metric inequalities are generated with a specialized and efficient solver for nonlinear multicommodity flow problems.

In many respects, our approach resembles the Benders partitioning scheme. The essential difference is that the integer programming solver is not used to produce the best

Average	L1-norm	L2-norm	L_∞-norm	L2-norm*
	Sun.tr instances			
h0 to h1	0.55	0.50	0.18	0.45
h1 to h2	0.12	0.20	0.07	0.03
h2 to h4	0.05	0.09	0.01	0.06
	Sun.dense instances			
h0 to h1	0.95	0.97	0.92	0.92
h1 to h2	0.66	0.74	0.38	0.57
h2 to h4	0.23	0.34	0.11	0.35

* ℓ_2 norm with $(y - y^*)^T(\bar{y} - y^*) \leq 0$.

Table 11: Fraction of CPU time spent in computing metric inequalities

integer point in the polyhedral relaxation of the set of feasible capacities, but an improving integer solution that is closest to the best known integer solution. The advantage of the method are threefold. The method is easy to implement (no need to construct sophisticated and specialized valid cuts); it produces good solutions; and, last but not least, it is very general and can be applied straightforwardly to other mixed integer programming problems.

Acknowledgements

The authors gratefully acknowledge the support of the "Fonds National Suisse de la Recherche", of the University of Chile, of the department of Management Studies of the University of Geneva and of CORE, Catholic University of Louvain.

References

[1] P. Avella, S. Mattia, and A. Sassano. Metric inequalities and the network loading problem. *Discrete Optimization*, 4:103–114, 2007.

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

[3] F. Babonneau and J.-P. Vial. ACCPM with a nonlinear constraint and an active set strategy to solve nonlinear multicommodity flow problems. Forthcoming, *Mathematical Programming*, 2007.

[4] F. Babonneau and J.-P. Vial. An efficient method to compute traffic assignment problems with elastic demands. Forthcoming *Transportation Science*, 2007.

[5] A. Balakrishnan, T.L. Magnanti and P. Mirchandani. Network design. In: M. Dell'Amico, F. Maffioli, S. Martello (Eds.), *Annotated Bibliographies in Combinatorial Optimization*, Wiley, 1997 (Chapter 18).

[6] J. F. Benders. Partitioning Procedures for Solving Mixed-Variables Programming Problems. *Numerische Mathematik* 4:238–252, 1962.

[7] D. Bienstock, S. Chopra, O. Gunluk, and C.-Y. Tsail. Minimum cost installation for multicommodity network flows. *Mathematical Programming*, 81:177–199, 1998.

[8] D. Bienstock and O. Gunluk. Capacitated network design: Polyhedral structure, and computation. *INFORMS Journal On Computing*, 8(3):243–259, 1996.

[9] S. Chopra, I. Gilboa, S.T. Sastry. Source sink flows with capacity installation in batches. *Discrete Applied Mathematics* 85:165–192, 1998.

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

[11] M. Iri. On an extension of the max-flow min-cut theorem to multicommodity flows. *Journal of the Operations Research Society of Japan*, 13:129–135, 1971.

[12] M. Lomonosov. Feasibility conditions for multiflow problems. *Discrete Mathematics*, 1982.

[13] T.L. Magnanti, P. Mirchandani and R. Vachani. Modeling and solving the two-facility capacitated network loading problem. *Operations Research*, 43(1):142–157, 1995.

[14] OBOE: the Oracle Based Optimization Engine, Available at `https://projects.coin-or.org`, 2007.

[15] K. Onaga and O. Kakusho. On feasibility conditions of multicommodity flows in networks. *Transactions on Circuit Theory*, CT-18, 4:425–429, 1971.