RAIRO. RECHERCHE OPÉRATIONNELLE

M. MINOUX

J. Y. SERREAULT

Subgradient optimization and large scale programming : an application to optimum multicommodity network synthesis with security constraints

RAIRO. Recherche opérationnelle, tome 15, n° 2 (1981), p. 185-203

<http://www.numdam.org/item?id=RO_1981__15_2_185_0>

R.A.I.R.O. Recherche opérationnelle/Operations Research
(vol. 15, n° 2, mai 1981, p. 185 à 203).

SUBGRADIENT OPTIMIZATION AND LARGE SCALE PROGRAMMING: AN APPLICATION TO OPTIMUM MULTICOMMODITY NETWORK SYNTHESIS WITH SECURITY CONSTRAINTS (*)

by M. MINOUX (1) and J. Y. SERREAULT (2)

Abstract. — *This paper is concerned with the solution of the following large scale optimization problem: given a N-node, M-arc connected graph* $G=[\mathcal{X}, \mathcal{U}]$, *determine a M-vector* $Y=(Y_u)_{u\in\mathcal{U}}$ *of capacities associated with the arcs, meeting any one of p given (independant) multicommodity requirements, and minimizing a linear cost function* $z=\sum_{u\in\mathcal{U}} \gamma_u Y_u$. *This problem has important applications in telecommunication network optimization and planning, when security constraints are imposed.*

As an alternative to linear programming techniques, it is shown how the use of Lagrangean relaxation for decomposition purposes, and the use of subgradient algorithms to optimize the dual problems, lead to a practical and efficient solution procedure.

Computational results obtained show that solutions within 5 to 10% of the optimum are easily obtained at low computational cost. This is especially interesting in view of the fact that even moderate sized problems (e.g. 50 nodes, 100 arcs, and $p=100$) *lead to very large scale linear programs (several hundreds of thousands variables and constraints in node-arc formulation) for which even the most sophisticated linear programming techniques could not provide exact solutions.*

An explanation of the apparent superiority of subgradient optimization over the simplex method for large scale programming is attempted in conclusion.

Keywords: Large scale linear programming, subgradient optimization, Benders decomposition, Lagrangean relaxation, network synthesis, multicommodity flows.

Résumé. — *Cet article se rattache au domaine de l'optimisation des grands systèmes et traite du problème suivant : étant donné un graphe connexe* $G=[\mathcal{X}, \mathcal{U}]$, *déterminer un vecteur* $Y=(Y_u)_{u\in\mathcal{U}}$ *de capacités associées aux arêtes permettant de satisfaire un quelconque de p multiflots (indépendants) donnés, et minimisant une fonction de coût linéaire* $z=\sum_{u\in\mathcal{U}} \gamma_u Y_u$. *Ce problème a d'importantes applications dans l'optimisation et la planification des réseaux de télécommunications lorsque l'on impose des contraintes de sécurité.*

(*) Received November 1979.
Paper presented at the Xth International Symposium on Mathematical Programming, Montreal, August 1979.

(1) Scientific advisor by The Centre National d'Études des Télécommunications and E.N.S.T.A., 32, boulevard Victor, 75015 Paris.

(2) Engineer at the Centre National d'Études des Télécommunications, 3, avenue de la République, 92130 Issy-les-Moulineaux.

R.A.I.R.O. Recherche opérationnelle/Operations Research, 0399-0559/1981/185/$ 5.00

Comme alternative aux techniques de la programmation linéaire, on montre comment l'utilisation de la relaxation lagrangienne, qui permet la décomposition du problème, et l'emploi d'algorithmes de sous-gradients pour optimiser le problème dual, conduisent à une méthode de résolution pratique et efficace.

Les résultats de calcul obtenus montrent que de solutions à moins de 5 à 10 % de l'optimum sont obtenues dans des temps de calcul très modestes. Ceci est particulièrement intéressant si l'on considère que même des graphes de taille modérée (par ex : 50 sommets, 100 arêtes, $p=100$) conduisent à des programmes linéaires de très grandes dimensions (plusieurs centaines de milliers de variables et de contraintes en formulation arcs-sommets) qui ne pourraient être résolus de façon exacte par les techniques de la programmation linéaire, même les plus élaborées.

Une tentative d'explication de la supériorité apparente des méthodes de sous-gradients sur la méthode du simplexe pour des programmes linéaires de grandes dimensions est suggérée en conclusion.

Mots-clés : Programmation linéaire de grandes dimensions, optimisation par sous-gradients, décomposition de Benders, relaxation lagrangienne, synthèse de réseaux multiflots.

1. PROBLEM STATEMENT AND FORMULATION

In the area of Telecommunication network design and planning, the problem of designing a minimum cost network meeting multicommodity requirements *under security constraints* is a fundamental one.

It has been shown in [2] and [3] that this problem can be formulated as follows:

Given a N-nodes, M-edges unoriented multigraph $G=[\mathscr{X}, \mathscr{U}]$, determine a M-vector of capacities $Y=(Y_u)_{u\in\mathscr{U}}$ assigned to the edges such that:

1) the capacities $Y=(Y_u)$ are feasible for any one of p (given) multicommodity flow requirements, $d^1, d^2, \ldots, d^p$;

2) the total cost $z=\sum_{u\in\mathscr{U}} \gamma_u Y_u$ is minimized, where γ_u denotes the cost of one unit of capacity on edge u.

This problem may be viewed as a generalization of the optimum network synthesis problem treated by Gomory and Hu (1962) in the single-commodity case.

Physically, each of the p given multicommodity flows represents the requirements to be restored in one possible failure configuration of the network. Thus, p is the number of failure configurations, and the network has to be dimensionned so as to operate under any one failure configuration (for more details about the physical problem, *see* [8]).

In practice, it should be noted that the (unknown) capacities Y_u are often constrained to be integers. However, we observe that: 1) since very large scale linear programs are involved, the quest for exact optimal solutions in integers is hopeless; 2) there exist simple and efficient heuristic procedures (*see* [8]) for converting an optimal continuous solution into an integer solution with very little additional cost (hence, very close to optimality). For these reasons, we shall restrict ourselves here to the *continuous problem*, i.e. with the integrality conditions relaxed.

Now, the problem to be solved may equivalently be expressed as:

$$(P_0)\quad \begin{cases} \text{Minimize:} \\ \qquad \gamma\, Y = \sum_{u \in \mathcal{U}} \gamma_u\, Y_u, \\ \text{under the constraints:} \\ \qquad Y \in \mathcal{D}^r \qquad (r = 1, \ldots, p), \\ \qquad Y \in \mathbb{R}^{M+}, \end{cases}$$

where, for each $r = 1, \ldots, p$, $\mathcal{D}^r$ is the convex (unbounded) polytope of $\mathbb{R}^M$ such that $Y \in \mathcal{D}^r$ if, and only if, there exists a multicommodity flow ψ^r meeting the d^r requirements, and feasible for Y, that is to say: $\psi^r \leqq Y$.

Using a node-arc formulation for the feasible multicommodity flow problem (*see* [1], chap. 6 for instance) each polytope $\mathcal{D}^r$ may be seen as the solution set of a linear program with $2.M.N.$ variables and N^2 constraints. With this representation, problem (P_0) can be formulated as a linear program with about $2p.M.N.$ variables and pN^2 constraints. For typical values of N, M, p ($N = 50$, $M = 100$, $p = 100$) we find 10^6 variables and 250 000 constraints: we are thus confronted to very large scale linear programs, for which there is no hope of getting exact solutions, even by means of the most sophisticated linear programming techniques.

Our purpose, here, is to show how subgradient optimization, used in conjunction with lagrangean relaxation and decomposition techniques, has been successfully used to cope with such very large scale linear programming problems.

2. NECESSARY AND SUFFICIENT CONDITIONS FOR FEASIBLE MULTICOMMODITY FLOWS

Instead of a node-arc formulation, another useful representation of the $\mathcal{D}^r$ polytope will be used.

It can be shown, from duality theory (*see* [1], chap. 6 for instance) that $Y \in \mathcal{D}^r$ if and only if the following condition ($\mathcal{C}$) holds:

$\mathcal{C}$ For any M-vector $\pi = (\pi_u) \geqq 0$ (π_u may be seen as a "cost" assigned to edge $u \in \mathcal{U}$):

$$\pi\,.\,Y = \sum_{u \in \mathcal{U}} \pi_u\,.\,Y_u \geqq \theta^r(\pi), \tag{1}$$

where $\theta^r(\pi)$ is the cost of the minimum cost network meeting the (multicommodity) requirement d^r, with respect to the π_u.

Note that $\theta^r(\pi)$ is easily obtained by assigning every component d^r_{ij} of d^r (requirement between nodes i and j) to the minimum cost chain between i and j in G relative to the $\pi_u (u \in \mathscr{U})$. Thus, obtaining $\theta^r(\pi)$ in (1) reduces to shortest path computations, a well solved problem in graph theory.

Condition ($\mathscr{C}$) can also be reformulated as a maximization problem. Observe that, if $\pi \geqq 0$ satisfies (1), so does $\lambda . \pi$ ($\forall \lambda \in \mathbb{R}^+$) and thus, it is not restrictive to impose a normalization condition, e. g.:

$$\pi . \mathbf{1} = \sum_{u \in \mathscr{U}} \pi_u = 1,$$

($\mathbf{1}$ denotes the M-vector with all components $= 1$).

Testing condition ($\mathscr{C}$) then appears to be equivalent to:

$$(FP) \qquad \begin{cases} \text{Max} \quad F(\pi) = \theta^r(\pi) - \pi . Y, \\ \text{subject to:} \\ \qquad \pi . \mathbf{1} = 1; \qquad \pi \geqq 0 \end{cases}$$

(feasibility problem).

Let $F^* = F(\pi^*)$ be an optimal solution of (FP):
if $F^* \leqq 0$, then ($\mathscr{C}$) holds and $Y \in \mathscr{D}^r$;
if $F^* > 0$, then ($\mathscr{C}$) does not hold, since for π^*:

$$\pi^* . Y < \theta^r(\pi^*).$$

Problem (FP) may be solved by generalized linear programming (i. e.: column generation techniques), but a more efficient approach (first suggested in [5] for the maximum multicommodity flow problem) is to use a subgradient algorithm (*see* section 6).

3. NEW FORMULATION OF (P_0) AND A CONSTRAINT GENERATION ALGORITHM

It may be shown (*see* [2]) that, in order to check condition ($\mathscr{C}$), only a finite (though very large) number of π vectors need be considered (they correspond to the extreme points of some convex polytope).

For multicommodity r, these vectors will be denoted by:

$$\pi^{r,1}, \pi^{r,2}, \ldots, \pi^{r,\alpha_r}.$$

For simplicity, we note $\theta^{r,1}, \theta^{r,2}, \ldots, \theta^{r,\alpha_r}$ the corresponding values of $\theta^r(\pi)$.

We now obtain a new representation of polytope $\mathscr{D}^r$ by means of the following system of linear inequalities:

$$Y \in \mathscr{D}^r \Leftrightarrow \begin{cases} \pi^{r,1} . Y \geqq \theta^{r,1}, \\ \pi^{r,2} . Y \geqq \theta^{r,2}, \\ \vdots \\ \pi^{r,\alpha_r} . Y \geqq \theta^{r,\alpha_r}, \\ Y \geqq 0. \end{cases}$$

It follows that problem (P_0) may be rewritten as:

$$(P_1) \quad \begin{cases} \text{Min } \gamma Y, \\ \text{subject to:} \\ \pi^{r,j} . Y \geqq \theta^{r,j} \qquad (r = 1, \ldots, p; j = 1, \ldots, \alpha_r), \\ Y \geqq 0. \end{cases}$$

Of course, (P_1) cannot be written explicitely, due to the enormous number of constraints. The following constraint generation algorithm, closely related to the Benders decomposition technique (*cf.* [15], and [17], chap. 7), had already been suggested in [2] and [3] for solving this problem.

ALGORITHM 1 (constraint generation algorithm):

(a) *At the current iteration, solve a* restricted problem (PR) *consisting in only a few constraints of* $(P1)$. *Let* $\bar{Y}$ *be an optimal solution of* (PR).

(b) *For* $r = 1, 2, \ldots, p$:

Solve problem (FP) *with* $Y = \bar{Y}$ *to check whether* $\bar{Y} \in \mathscr{D}^r$. *If the answer is "yes"* $(F^* \leqq 0)$ *then proceed to next r. If "no"* $(F^* \geqq 0)$ *then the constraint:* $\pi^* . Y \geqq \theta^r(\pi^*)$, *violated by the current* $\bar{Y}$, *is added to the current restricted problem, and we proceed to next r.*

(c) *Two situations may occur:*

(i) *no extra constraint has been added in step* (*b*): *the current* $\bar{Y}$ *is an optimal solution of* (P_1) *and* (P_0), *and the algorithm terminates;*

(ii) *some new constraints have been added in step* (*b*); *thus forming a new restricted problem.*

Return to (*a*).

Finite convergence of algorithm 1 was proved in [2], assuming that the feasibility problem (FP) was solved exactly at each iteration.

Such a result, however, is mainly of theoretical interest. In practice, we are much more interested in a good rate of convergence (even if not finite) rather than achieving a finite (possibly slow) convergence. The following sections will now be devoted to the question of obtaining an efficient implementation (in the above sense) of algorithm 1. For that, the main problems to be discussed are:

(*a*) how to start the iterations;

(*b*) how to solve the restricted problem at each iteration;

(*c*) how to solve the p multicommodity feasibility problems (FP) at each iteration;

(*d*) when to stop the algorithm.

It will be shown in particular how subgradient optimization can be used to solve these problems efficiently and thus provide good overall convergence characteristics.

4. INITIALIZATION OF ALGORITHM 1 BY SOLVING A DUAL PROBLEM

Obviously, the key point in the efficiency of algorithm 1 is the choice of the starting restricted problem; in other words, which constraints should be selected out of (P_1) to start the iterations?

The main idea, here, is to solve a dual problem of (P_0) obtained via Lagrangean relaxation of the coupling constraints. The optimal (or suboptimal) dual variables obtained are then used to build the starting restricted problem (PR_0). A nice feature of the method, is that it is not necessary to solve the dual exactly: good approximate solutions are sufficient. It also provides tight lower bounds of the optimal value of the primal problem (P_0).

Fist, we observe that, introducing the vectors of auxiliary variables X^r, problem (P_0) may be equivalently written as:

$$(P'_0) \qquad \begin{cases} \text{Min } \gamma\ Y, \\ \text{subject to:} \\ X^r \leqq Y \qquad (r=1, \ldots, p), \\ X^r \in \mathscr{D}^r \qquad (r=1, \ldots, p), \\ Y \in \mathbb{R}^{M+}, \end{cases} \tag{2}$$

where $X^r=(X^r_u)_{u\in\mathscr{U}}$, and X^r_u is the actual total flow of multicommodity r through arc u in the network.

Now, let us associate with the coupling constraints (2) a Mxp-vector $\rho=(\rho_u^r)\,(r=1,\ \ldots,\ p)\,(u=1,\ \ldots,\ M)$ of *Lagrange multipliers* $(\rho \geqq 0)$ and consider the *dual function:*

$$L(\rho)=\min_{\substack{X^r\in\mathscr{D}^r\\ Y\in\mathbb{R}^{M+}}}\left[\gamma\, Y+\sum_{r=1}^{P}\rho^r\,(X^r-Y)\right],$$

where $\rho^r=(\rho_u^r)_{u=1,\ \ldots,\ M}$.

We note that $L(\rho)$ readily *decomposes* into a sum of $p+1$ terms:

$$\mathrm{L}(\rho)=\min_{Y\in\mathbb{R}^{M+}}\left\{\left[\gamma-\sum_{r=1}^{P}\rho^r\right].Y\right\}+\sum_{r=1}^{P}\min_{X^r\in\mathscr{D}^r}\rho^r.X^r.$$

The first term may be written as:

$$\sum_{u=1}^{M}\left(\gamma_u-\sum_{r=1}^{P}\rho_u^r\right).Y_u,$$

and any bounded minimum is reached for $Y_u=0$ provided that:

$$\sum_{r=1}^{P}\rho_u^r\leqq\gamma_u,\qquad\forall u\in\mathscr{U}. \tag{3}$$

If condition (3) is imposed (which will be assumed later on) the first term reduces to 0. The following terms in $L(\rho)$ are computed by (independently) solving p problems of the form:

$$(Q^r)\qquad\begin{cases}\text{Min } \rho^r.X^r,\\ \text{subject to:}\\ X^r\in\mathscr{D}^r.\end{cases}$$

We notice that the solution of (Q^r) is nothing but the minimum cost network meeting the d^r requirements, the costs $\rho_u^r\,(u\in\mathscr{U})$ being assigned to the edges of G. Thus, each (Q^r) is solved by means of shortest path computations (*see* § 2) and the cost of an optimal solution $\bar{X}^r$ of (Q^r) is exactly $\theta^r(\rho^r)$ previously defined in section 2.

Since $L(\rho)$ can be efficiently computed for any value of ρ [$L(\rho)=-\infty$ if ρ does not satisfy (3)], the dual problem (D) of (P_0') can be stated as:

$$(D)\qquad\begin{cases}\text{Maximize } L(\rho),\\ \rho\geqq 0.\end{cases}$$

We recall the following well-known results in Lagrangean duality (*see* [10] or [1] appendix 3 for instance):

1) $L(\rho)$ is a *concave* piecewise linear function;

2) $$\forall \rho \geqq 0 : L(\rho) \leqq L(\rho^*) = \gamma . Y^*,$$

where Y^* is an optimal solution of (P_0) or (P'_0) and ρ^* is an optimal solution of (D) (lower bound property);

3) for any $\rho \geqq 0$ satisfying (3) :

$$t(\rho) = (\bar{X}^1, \bar{X}^2, \ldots, \bar{X}^p)$$

is a *subgradient* of L at ρ [$\bar{X}^r$ denotes an optimal solution of (Q^r)].

It follows that the dual problem (D) can be solved by means of a *subgradient algorithm* of the following type:

PROCEDURE 1 (solving the dual problem):

(a) *Step* 0: $\rho = \rho^0$ (e. g. $\rho^0 = 0$).

(b) *Step j: ρ^j is the current solution.*

Compute $L(\rho^j)$ and $t(\rho^j)$, a subgradient of L at ρ^j.

(c) *Let: $\rho' = \rho^j + \lambda_j t(\rho^j)$ (λ_j is a given step size). If ρ' violates some constraints* (3), *project ρ' on the corresponding hyperplanes, thus obtaining ρ''. Define ρ^{j+1} as the projection of ρ'' on $\mathbb{R}^{M+}$.*

Set $j \leftarrow j+1$, and return to (b), *or terminate as soon as a stop condition is fulfilled.*

There are many possible choices for the step sizes λ_j (*see* [1] appendix 3, [5], [11] for instance) and for the stop criterion. The strategy used for solving the examples of section 8 is the following (*see* [4]):

$$\lambda_j = \alpha_j \frac{\hat{L} - L(\rho^j)}{\| t(\rho^j) \|^2},$$

where:

– $\hat{L}$ is an upper bound of $L(\rho^*)$, derived from a good approximate solution obtained by a heuristic procedure applied to (P_0) (a class of such heuristic procedures is described in [8]).

– α_j is a sequence of real numbers ($0 < \alpha_j \leqq 2$) defined by a rule of the following type [4]:

(*a*) choose α_0, K_0, N_0 (in the examples of § 8), $\alpha_0 = 2$, $K_0 = 200$, $N_0 = 30$. Set $\alpha = \alpha_0$, $K = K_0$; (*b*) perform Kiterations with $\alpha_j = \alpha$; (*c*) set $K \leftarrow \max(K/2, N_0)$ and $\alpha \leftarrow \alpha/2$ and return to (*b*).

The stop condition may be allowing a maximum number of iterations, or testing whether the step size becomes smaller than a given tolerance $\lambda_{\min}$ (*see* section 6).

Now, let $\bar{\rho}=(\bar{\rho}^r)$ be the vector of Lagrange multipliers (dual variables) corresponding to the best solution obtained by procedure 1 and consider the following restricted problem of (P_1), the p constraints of which correspond to $\bar{\rho}^1$, $\bar{\rho}^2, \ldots, \bar{\rho}^p$ respectively:

$$(PR_0) \quad \left\{ \begin{array}{l} \text{Min } \gamma\, Y, \\ \text{Subject to:} \\ \bar{\rho}^1 . Y \geqq \theta^1(\bar{\rho}^1), \\ \bar{\rho}^2 . Y \geqq \theta^2(\bar{\rho}^2), \\ \vdots \\ \bar{\rho}^p . Y \geqq \theta^p(\bar{\rho}^p), \\ Y \geqq 0. \end{array} \right.$$

Then, it can be shown that an optimal solution $\bar{Y}$ of (PR_0) satisfies:

$$L(\bar{\rho}) \leqq \gamma\, \bar{Y} \leqq \gamma\, Y^* = L(\rho^*), \tag{4}$$

(*see* proof in [8]).

In other words, starting algorithm 1 with (PR_0) will produce a solution $\bar{Y}$ with cost at least as close to the optimum cost as is $L(\bar{\rho})$, the best lower bound obtained by solving the dual (D).

Since $L(\rho^*)$ can be approximated as closely as desired by $L(\bar{\rho})$ – the accuracy depending on the number of iterations in procedure 1 – we conclude that solving the dual provides a systematic way of getting good starting solutions for algorithm 1.

Notice that, though $\gamma . \bar{Y}$ is very close to $\gamma . \bar{Y}^*$, $\bar{Y}$ may be quite different from Y^*, thus iterations of *Algorithm* 1 are necessary anyway.

5. SOLVING THE RESTRICTED PROBLEM

At each iteration of algorithm 1, a restricted problem of the form:

$$(PR) \quad \left\{ \begin{array}{l} \gamma\, \bar{Y} = \min \gamma\, Y, \\ \text{subject to:} \\ \pi^i . Y \geqq \theta^i \qquad (i=1, \ldots, q), \\ Y \geqq 0, \end{array} \right.$$

has to be solved.

Clearly, this could be done by the simplex method, but we found it better to solve (*PR*) approximately by means of a relaxation scheme [12, 13] closely related to subgradient optimization as shown in [5].

In fact, we notice that, each time (*PR*) must be solved, a good lower bound $\bar{z}$ of $\gamma\,\bar{Y}$ is known: at the first iteration, $\bar{z}=L(\bar{\rho})$; for subsequent iterations, $\bar{z}$ may simply be taken as the optimal value of the previous restricted problem.

(*PR*) is then equivalent to:

$$\begin{cases} \text{Minimize } \varepsilon, \\ \text{subject to:} \\ \gamma\,.\,Y \leqq \bar{z}(1+\varepsilon), \\ \pi^i\,.\,Y \geqq \theta^i \qquad (i=1,\ \ldots,\ q), \\ Y \geqq 0, \end{cases}$$

and an efficient (and easy-to-implement) way of finding the minimum ε^* is the following iterative procedure:

PROCEDURE 2 (solving the restricted problem):

(a) $\varepsilon=\varepsilon^0$ (*for example if* 20% *is an estimation of how close* $\bar{z}$ *approximates* $\gamma\,.\,\bar{Y}$, $\varepsilon^0=0.2$).

(*b*) *Look for Y satisfying the system of linear inequalities:*

$$\text{(I)} \qquad \begin{cases} \gamma\,Y-\bar{z}(1+\varepsilon)\leqq 0, \\ -\pi^i\,.\,Y+\theta^i\leqq 0 \qquad (i=1,\ \ldots,\ q), \\ Y\geqq 0, \end{cases}$$

using a fixed number of steps of a relaxation procedure (*e. g.* = 5 *to* 10 *steps*). *The starting point is taken as the last feasible solution obtained.*

(*c*) *If a solution of* (*I*) *has been found, then decrease* ε (*e. g:* $\varepsilon \leftarrow \varepsilon/2$) *and return to* (b); *otherwise increase* ε (*e.g.*: $\varepsilon \leftarrow (3/2)\varepsilon$) *and return to* (*b*).

The choice of this relaxation scheme, instead of the simplex method, has been motivated mainly by its low memory requirements, and also by its easy implementation. A drawback of the method might have been that it only provides approximate solutions (though the accuracy can be made as high as desired by increasing the number of iterations). In practice, however, it has been observed that solutions sufficiently close to optimality could be obtained without much computational effort, and thus the overall convergence of algorithm 1 was not affected by the approximation. This is why it didn't appear necessary to use the (dual) simplex method which, at first sight, could be thought of as a "more natural" approach. Keep also in mind that the chief argument in favor of the simplex method—getting exact solutions in a finite number of steps—is only

of theoretical value: in practice, due to round-off errors, only approximate solutions are obtained (even cycling or premature termination can occur), and this is especially true for large scale problems.

6. SOLVING THE FEASIBILITY PROBLEM (*FP*)

Once the current solution $\overline{Y}$ of (PR) has been obtained, we have to check whether $\overline{Y} \in \mathscr{D}^r$ for $r = 1, \ldots, p$. In section 2, this problem was shown to be equivalent to:

$$(FP) \quad \left\{ \begin{array}{l} \text{Max} \quad F(\pi) = \theta^r(\pi) - \pi\,\overline{Y}, \\ \text{subject to:} \\ \pi.\mathbf{1} = 1, \qquad \pi \geqq 0. \end{array} \right.$$

It may be shown (*cf.* [1], chap. 6). That:

1) $F(\pi)$ is a piecewise linear concave function of π;

2) $F(\pi)$ is not everywhere differentiable, but for any $\pi \geqq 0$ $(\pi.\mathbf{1} = 1)$ a *subgradient* $\gamma(\pi)$ of F at π can easily be computed.

$\gamma(\pi)$ is obtained as follows: if $X_u(\pi)$ denotes the total flow through edge u on the network of minimum cost $\theta^r(\pi)$ (*see* § 2), then $\gamma(\pi) = X(\pi) - \overline{Y}$ Hence, $\gamma(\pi)$ is obtained as a by-product of the shortest path computations performed for getting $\theta^r(\pi)$.

It follows that each feasibility problem (*FP*) may be solved with a subgradient algorithm such as:

PROCEDURE 3 (solving the feasibility problem):

(a) π^0 *is the starting point (for instance* $\pi^0 = 0$).

(b) *At step* k, π^k *is the current solution.*

Compute $\theta^r(\pi^k)$ *and* $\gamma(\pi^k)$ *using a shortest path algorithm.*

(c) *Define* $\pi' = \pi^k + \lambda_k.\gamma(\pi^k)$ (λ_k *is a step size*).

Project π' *on* $\mathbb{R}^{M+}$ (i.e. : *whenever* $\pi'_u < 0$, *set* $\pi'_u = 0$), *and define:*

$$\pi^{k+1} = \pi'/\pi'.\mathbf{1}.$$

Set $k \leftarrow k+1$ *and go to step* (b).

Just as for *procedure* 1, various strategies for the choice of the step sizes λ_k have been studied (*see* [1], appendix 3). One possible choice [14] is:

$$\lambda_k = \lambda_0 \frac{(\sigma)^k}{\| \gamma(\pi^k) \|},$$

where λ_0 is the initial step and $0 < \sigma < 1$. For a proper choice of λ and σ, *see* [11].

The use of a subgradient algorithm in procedure 3 is justified by the fact it is not necessary to solve (FP) exactly for getting a good approximate solution of the global problem (P_0). As will be shown in section 7 below, testing ε-feasibility is sufficient, and thus, it is easy to see that the iterations in procedure 3 may be stopped as soon as:

$$F(\pi^*) - F(\pi^k) \leqq \varepsilon,$$

[π^* is an optimal solution of (FP)], which holds if:

$$\| \gamma(\pi^k) \| . \| \pi^* - \pi^k \| \leqq \varepsilon$$

i.e. :

$$\| \pi^* - \pi^k \| \leqq \frac{\varepsilon}{\| \gamma(\pi^k) \|}.$$

Now, testing this condition generally reduces to checking whether the step size λ_k becomes smaller than some threshold value. For instance, if the λ_k are computed according to the rule:

$$\lambda_k = \lambda_0 \frac{(\sigma)^k}{\| \gamma(\pi^k) \|},$$

and assuming that λ_0 and σ are chosen in such a way that: $\pi^k \to \pi^*$. Then:

$$\| \pi^* - \pi^k \| \leqq \lambda_0 \sigma^k (1 + \sigma + \sigma^2 + \ldots) = \frac{\lambda_0 (\sigma)^k}{1 - \sigma}.$$

In this case, procedure 3 will be stopped at iteration k when:

$$\frac{\lambda_0 (\sigma)^k}{1 - \sigma} \leqq \frac{\varepsilon}{\| \gamma(\pi^k) \|} < \frac{\lambda_0 (\sigma)^{k-1}}{1 - \sigma}.$$

If the ε-optimal solution obtained is such that $F(\pi^k) > 0$, then it is used to generate a new constraint:

$$\pi^k . Y \geqq \theta(\pi^k),$$

which is added to the current restricted problem (PR). Observe that the constraints thus generated are not necessarily constraints of (P_1), though it is easy to show that they are convex combinations of constraints of (P_1).

7. STOPPING THE ITERATIONS

If the feasibility problems (FP) were solved exactly at each iteration, finite convergence of algorithm 1 can be proved (*see* [2]). Clearly this finite convergence property does not hold any more when an approximate method like procedure 3 is used, and a stop criterion is necessary.

We say that $Y \in \mathbb{R}^{M+}$ is ε-*feasible* with respect to multicommodity r if $Y+\varepsilon.\mathbf{1} \in \mathscr{D}^r$. In other words, by adding ε to every $Y_u (u \in \mathscr{U})$ a feasible multicommodity flow can be found. For testing ε-feasibility of the current solution in algorithm 1 a good approximate algorithm is sufficient.

For example, procedure 3 with a stop criterion ensuring ε-optimality, as discussed in section 6, is well suited. The "feasibility program" described in [6] and [16] can also be used, and offers the additional advantage of providing a primal solution.

A possible stop criterion is then the following: given ε, stop at step (b) of algorithm 1 as soon as the current solution $\overline{Y}$ is ε-feasible for each multicommodity $r = 1, \ldots, p$.

Clearly, then, $\overline{Y}+\varepsilon.\mathbf{1}$ is a solution of (P_0), thus $\gamma.(\overline{Y}+\varepsilon\,\mathbf{1})$ is an upper bound of the optimum cost $\gamma.Y^*$, and we have:

$$\gamma.\overline{Y} \leqq \gamma.Y^* \leqq \gamma(\overline{Y}+\varepsilon.\mathbf{1}).$$

Hence, for ε small enough, the algorithm will terminate with a good approximate solution.

Computational experience (section 8) shows that for $\varepsilon \simeq 5\,\%$, no more than 4 to 8 iterations of algorithm 1 are necessary to terminate. Such a fast convergence seems primarily due to the quality of the starting restricted problems obtained from procedure 1.

8. COMPUTATIONAL RESULTS

The numerical experiments presented here concern the basic network shown in figure 1 with $N = 12$ nodes and $M = 25$ edges, on which are routed, in normal operations, 66 distinct point-to-point requirements. We consider here six different problems. For instance, in problem 1 every requirement is routed on one single path (unirouting) on the basic network; in problem 2 every requirement is split into two equal parts routed on two edge-disjoint paths on the basic network (birouting). Problems 3 to 6 have been built in a similar way.

For each problem, the p multicommodity-flows to be considered are obtained as follows. One failure configuration consists in the total breakdown of any one edge of the basic network. Now, to each failure configuration (to each edge) corresponds a specific multicommodity flow composed of all the point-to-point requirements that were passing through the broken edge in the basic network. Thus, problems 1 to 6 consist in $p = M = 25$ non-simultaneous multicommodity flows. Finally, all the costs $\gamma.Y$ are expressed in percentage of the cost of the basic unirouted network.

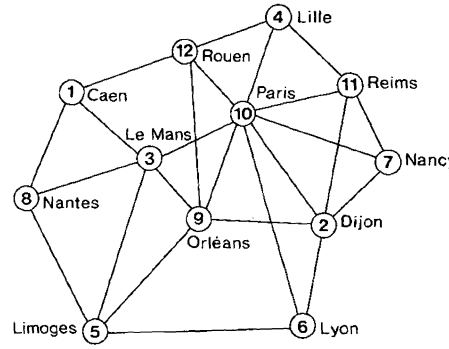

Figure 1. – Test network

We give for problems 1 and 2:

1) The sequence of lower bounds obtained by procedure 1 (tables I and IV).

TABLE I

Problem 1: sequence of lower-bounds.

j	10	100	200	300	400
$L(\rho^j)(\%)$. . . .	7	21	27	32	35

N.B.: $L(\rho)$ here is a function of $M(M-1)=600$ variables.

2) The solutions obtained after solving the successive restricted problems (PR) (tables II and V). Only the first five components of the solutions $\bar{Y}$ of (PR) are shown.

TABLE II

Problem 1: solving the restricted problem (PR)
[N.B. Q = number of constraints of (PR)]

Iteration	Q	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}_5$	$\gamma.\bar{Y}(\%)$
0.	25	88.6	89.4	44.4	60.4	61.3	35.0
1.	35	96.2	79.3	44.7	107.3	70.2	35.5
2.	42	104.0	74.6	57.9	120.5	62.9	35.5
3.	49	98.5	84.0	57.9	110.6	67.1	35.7
4.	52	103.8	83.2	60.3	111.7	70.0	35.2
5.	55	104.3	83.9	61.0	113.6	69.9	35.9
6.	58	106.3	81.3	60.1	116.0	70.1	35.2

3) The first five components of the integer solution obtained with the heuristic method of reference [8], and its cost which is an upper bound of both the continuous and integer optimums (tables III and VI) ([3]).

TABLE III

Problem 1: integer upper-bound via an approximate algorithm

Y_1	Y_2	Y_3	Y_4	Y_5	$\gamma . Y(\%)$
124	83	65	117	75	36.9

Less detailed results concerning problems 3 to 6 have been summarized in table VII.

TABLE IV

Problem 2: sequence of lower bounds

j	10 (%)	20 (%)	30 (%)	40 (%)	50 (%)	60 (%)	70 (%)
$L(\rho^j)$.	−2	4	12	18	21	22	22.8

We observe that, in most cases, solutions very close to the exact continuous optimum are obtained. The maximum relative error is $(36{,}9-35{,}2)/35{,}2=5\%$ for problem 1 and $(31{,}9-31{,}5)/31{,}5=1{,}3\%$ for problem 2.

TABLE V

Problem 2: solving the restricted problem (PR)

i	Q	$\bar{Y}_1$	$\bar{Y}_2$	$\bar{Y}_3$	$\bar{Y}_4$	$\bar{Y}_5$	$\gamma \bar{Y}(\%)$
0.	25	0.2	28.9	121.8	0.1	93.3	26.6
1.	46	2.4	71.2	64.5	69.5	6.0	27.6
2.	61	46.5	74.7	111.9	48.3	55.2	31.4
3.	74	30.9	79.4	82.8	33.3	61.5	28.2
4.	83	28.6	69.3	93.0	42.2	62.2	31.5

([3]) Comparing the cost of the solution obtained with the cost of a feasible integer solution (which is an upper bound of the cost of a continuous optimum solution) may lead to underestimate the actual quality of the results. Thus, it may be asked why we did not perform the evaluation by simply comparing with the exact continuous optimum solution. The answer is straightforward: getting such a solution is the very problem that we originally wanted to solve The motivation of our work lies precisely in the observation that, even for networks of rather moderate size (like those studied here) exact solutions were very difficult, if not impossible, to obtain by classical linear programming techniques. In contrast, approximate integer solutions are rather easy to obtain via heuristic procedures, as those described in [8].

Moreover, it is seen that very few main iterations of algorithm 1 are necessary and that the total number of constraints added to the starting restricted problem is thus quite moderate.

TABLE VI

Problem 2: integer upper-bound via an approximate algorithm

Y_1	Y_2	Y_3	Y_4	Y_5	$\gamma\, Y(\%)$
34	73	103	44	68	31.9

The algorithm has been implemented on a HB 6080 computer. For the various problems tested on the network of figure 1, the program needed about 40 K words of memory and about 20 minutes of processing time.

TABLE VII

	Procedure 1		Algorithm 1			Approximate algorithm upper bound (integer solution)	Relative error (%)
	Lower bound obtained	Number of iterations	Number of iterations	Total number of constraints (PR)	Cost of final solutions (PR)		
Problem 3.....	43,6	250	4	72	44,9	49,6	9
Problem 4.....	46,8	250	4	76	47,5	55,8	17
Problem 5.....	36,0	250	4	69	37,3	41,7	12
Problem 6.....	32,6	250	4	71	34,6	37,6	9

It should be noted that by now, this program is far from achieving the best possible implementation. Such possibilities as using the dual variables obtained at iteration k as starting solution for solving the p feasibility problems (FP) at iteration $k+1$ have not yet been used.

An improved version is presently beeing studied, and significant savings in the computation times are expected.

9. CONCLUSION: ON THE EFFICIENCY OF SUBGRADIENT OPTIMIZATION APPLIED TO LARGE SCALE PROBLEMS

The practical efficiency of subgradient optimization for solving large scale problems was first put into evidence by the pioneering work of Held and Karp [4] on large travelling salesman problems and in a subsequent paper [5] where the

method was applied to other problems like assignment and feasible multicommodity flows. Since that time, it has been successfully used to cope with a number of other important problems, such as: minimum cost multicommodity flows (Kennington and Shalaby, [19]) generalized assignment problems (Legendre and Minoux, [18]), etc.

The problem and the results presented here which, as far as we know, seems to be the first application of a subgradient technique in the context of Benders decomposition, further confirm the pratical interest of the method for treating large scale problems.

Though theoretical analysis has not fully succeded in explaining this efficiency yet, some efforts have been made to get some understanding of it. The work of Goffin [11] throws some light on the subject. The convergence of a subgradient algorithm applied to a convex (or concave) function can be shown to be *linear*, and he shows that the maximum sustainable rate of convergence σ is related to the so-called *condition number* χ of the function ($\sigma = \sqrt{1-\chi^2}$). Briefly, χ is the cosine of the largest angle between a subgradient at any point and the direction to the closest point in the (convex) set S of optimum points. With some additional assumption, it can be shown that the distance between the current solution and the optimum set S decreases as fast as σ^k (k is the iteration number) (*see* [11] for proofs). For instance, with a condition number $\chi = 0.3$ we get $\sigma \simeq 0.95$ and reducing the distance to the optimum set S by a factor 1 000 requires about 120 iterations. These results are very useful to understand why subgradient optimization may be superior to linear programming. First, let us try to estimate how the computational effort required by the simplex method increases with the size of the problem. Suppose that worst-case problems (i. e. for which running time grows exponentially with the size) are pathological ones, and that we only deal with average-case problems. According to statistical observations, the average number of iterations roughly appears to be proportional to the size of the problem measured by the number of constraints M. Since the computational effort per iteration grows at least as M^2, an overall estimation of the *average* complexity of the simplex method is $O(M^3)$. This means that increasing the size by a factor 10 results in a factor 1 000 on the computation time.

Now, what about subgradient optimization? Suppose that, within a given class of problems, the condition number of the function to optimize can be considered as *constant* (independant of the size of the problem). From the above convergence results, it follows that the number of iterations needed to get a given precision on the result (e.g. : reducing the initial distance by a factor 1,000) would be constant, and independant of M, the size of the problem. Under this

assumption, it is clear that for large scale problems (i.e. ; M large enough) subgradient optimization would be inherently superior to linear programming. Though the fact that the condition does not depend on the size is still a conjecture, the success of subgradient algorithms on many large scale problems, like the one studied here, tends to show, as already noticed in [11], that the associated condition numbers (surprisingly) remain rather high, and gives some support to the conjecture.

ACKNOWLEDGEMENTS

We tank the referee for his clever and constructive comments which helped to improve the original paper.

REFERENCES

1. M. GONDRAN and M. MINOUX, *Graphes et algorithmes*, Eyrolles, Paris, 1979.
2. M. MINOUX, *Optimum Synthesis of a Network with Non Simultaneous Multicommodity Flow Requirements*, Proceedings of the workshop "Application of Graph Theory and Combinatorics to management ", Brussels, March 20-21, 1979.
3. M. MINOUX, *La synthèse des réseaux de télécommunication avec contraintes de sécurité : une approche théorique*, Note technique ITD/CES 68 du Centre national d'Études des Télécommunications (non publiée), 1972.
4. M. HELD and R. M. KARP, *The Travelling Salesman Problem and Minimum Spanning Trees: part II*, Mathematical Programming, Vol. 1, 1971, pp. 6-25.
5. M. HELD, P. WOLFE and H. P. CROWDER, *Validation of Subgradient Optimization*, Mathematical Programming, Vol. 6, No. 1, 1974, pp. 62-88.
6. M. MINOUX, *Planification à court et à moyen terme d'un réseau de télécommunications*, Annales des Télécommunications, T. 29, No. 11-12, 1974, pp. 509-536.
7. M. MINOUX and J. Y. SERREAULT, *Le programme d'admissibilité avec contraintes de sécurité et coûts de mutations*, Annales des Télécommunications. 1980 (à paraître).
8. M. MINOUX and J. Y. SERREAULT, *Synthèse optimale d'un réseau de télécommunications avec contraintes de sécurité*, Annales des Télécommunications, 1981 (à paraître).
9. J. Y. SERREAULT, *Calcul automatique des plans de secours dans le réseau interurbain français : présentation du programme ORSEC*, Note technique du Centre national d'Études des Télécommunications, 1979 (non publié).
10. M. MINOUX *Programmation mathématique*, Cours de l'École nationale supérieure de Techniques avancées, Paris, 1978.
11. J. L. GOFFIN, *On Convergence Rates of Subgradient Optimization Methods*, Mathematical Programming, Vol. 13, 1977, pp. 329-347.
12. S. AGMON, *The Relaxation Method for Linear Inequalities*, Can. J. Math., Vol. 6, 1954, pp. 382-392.
13. T. MOTZKIN and I. J. SCHOENBERG, *The Relaxation Method for Linear Inequalities*, Can. J. Math., Vol. 6, 1954, pp. 393-404.

14. N. Z. SHOR, *On the Rate of Convergence of the Generalized Gradient Method*, Kibernetika, Vol. 4, No. 3, 1968.
15. J. F. BENDERS, *Partitioning Procedures for Solving Mixed Variables Programming Problems*, Numerische Mathematik, 4, 1962, pp. 238-252.
16. M. MINOUX, *Résolution des problèmes de multiflots en nombres entiers dans les grands réseaux*, R.A.I.R.O. (France), Vol. 3, 1975, pp. 21-40.
17. L. S. LASDON, *Optimization Theory for Large Systems*, Macmillan series for operations research, 1970.
18. J. P. LEGENDRE and M. MINOUX, *Une application de la notion de dualité en programmation discrète : sélection et affectation optimales d'une flotte d'avions*, R.A.I.R.O., Vol. 11, No. 2, 1977, pp. 201-222.
19. J. L. KENNINGTON and M. SHALABY, *An Effective Subgradient Procedure for Minimal Cost Multicommodity Flow Problems*, Manag. Sc., Vol. 23, No. 9, 1977, pp. 994-1004.