LARGE SCALE NONLINEAR NETWORK OPTIMIZATION WITH LINEAR SIDE CONSTRAINTS

F. Javier Heredia & Narcís Nabona
Statistics and Operations Research Department
Universitat Politècnica de Catalunya
Pau Gargallo 5, 08028-Barcelona
e-mail : heredia@eio.upc.es

ABSTRACT

This paper describes the work done by the authors on the design of an specialized algorithm for solving the nonlinear network flow problem with linear side constraints. The algorithm has been designed as an active set method for large scale problems employing the superbasic variable set strategy of Murtagh & Saunders . Within the general framework of the active set method, the special structure of the network flow problem is exploited using an extension of the Kennington & Helgason algorithm for linear network flow problems with side constraints.

1. . Introduction

1.1. . Formulation of the problem

The problem to solve is the minimization of a nonlinear objective function (a cost function) whose variables are de values of the flows passing trough the capacitated arcs of an oriented network. We expect also this flows to satisfy a set of linear inequality constraints, called the *side constrains* . The mathematical expression of this problem is :

$$\begin{aligned} \min \quad & f(x) && (1)\\ \text{subject to :} \quad & Ax = r && (2)\\ & Tx \leq b && (3)\\ & l \leq x \leq u && (4) \end{aligned}$$

we will refer to this problem as the NNS problem. In this formulation we have that :

(1) is a nonlinear function $f: R^n \rightarrow R$. $f(x)$ is suposed to be twice continuously differenciable on the feasible set defined by the constraints (2) to (4). The variables $x \in R^n$ represent the values of the arc flows in the network.

(2) represents the *Network Equations*. Matrix $A \in R^{m \times n}$ is the *node-arc incidence matrix* and $r \in R^m$ is the supply/demand vector. This m equality constraints

modelize the conservation of flows when passing through the nodes of the oriented network associated with A.

(3) is a set of t side constrains . We supose that $t < m$.

(4) $l, u \in R^n$ are the upper and lower bounds assigned to the flows in each arc of the oriented network.

1.2. . The Nonlinear Network Flow problem

The nonlinear network flow problem consists on the same problem expressed by equations (1), eliminating the side constrains . Many researchers have focused their attention on this topic. The usual way to tackle the problem is to combine a data structure of the type proposed by Bradley Brown and Graves in [2] with the variant of the active set method with superbasic variable introduced by Murtagh & Saunders in [7]

It is well known that in the network flow problem the set of basic arcs form a rooted spanning tree . All the operations involving the basic matrix B and its inverse B^{-1} can be performed efficiently working with a reduced set of vectors that describe the spanning tree . The actualization of B^{-1} due to a change in the basic set of variables can be also easily carried out through the modification of the vectors describing the spanning tree (it must be noted that no factorizations of B are needed).

1.3. The Linear Network Flow Problem with Side Constraints

In this section we introduce the basic terminology, notation and strategy that will allow the side constrains to be treated within the general network flow problem . We follow basically the exposition presented by Kennington & Helgason in [6].

Let us focus our attention on the constraints of the NNS problem. We make the following assumptions about NNS :

1) the graph the oriented network arises from is connected.

2) matrix S has full row rank.

3) total supply equals total demand

4) the lower bounds are null ($l = (\mathbf{0})$).

Let $\bar{A}$ be the constraint matrix of NNS :

$$\bar{A} = \left(\begin{array}{c} A \\ \hline T \end{array} \right)$$

Since the matrix A has rank $m - 1$ In order to have a full row rank constraint matrix it is necessary to introduce a *root arc* connected to an arbitrary node, say node l. The new expression of matrix $\bar{A}$ is :

$$\bar{A} = \begin{pmatrix} A & e_l \\ T & 0 \end{pmatrix}$$

The next three propositions are essential for the characterization of the basis of $\bar{A}$. See [6] for its proof :

Proposition 1.: Every basis for $\bar{A}$ may be placed in the following form :

$$\bar{B} = \begin{matrix} m\{ \\ t\{ \end{matrix} \overbrace{\begin{pmatrix} B & C \\ D & F \end{pmatrix}}^{\begin{matrix} m & t \end{matrix}} \tag{5}$$

where B is a submatrix of $(A \quad e^l)$ and $\det(B) \neq 0$

Proposition 2.: If $\bar{B}$ is invertible and B is invertible , then $F - DB^{-1}C$ is invertible.

Proposition 3.: If $\bar{B}$ is invertible and B is invertible,then :

$$\bar{B}^{-1} = \begin{pmatrix} B^{-1} + B^{-1}CQ^{-1}DB^{-1} & -B^{-1}CQ^{-1} \\ -Q^{-1}DB^{-1} & Q^{-1} \end{pmatrix}$$

where matrix Q, called the *working basis* ,is $Q = F - DB^{-1}C$.

Proposition 3 is the key to the efficient solution of systems $\bar{B}y = z$ and $y'\bar{B} = z'$. Consider the vectors y and z partitioned in the following way :

$$y = \begin{matrix} m\{ \\ t\{ \end{matrix} \begin{pmatrix} y^1 \\ y^2 \end{pmatrix} \qquad ; \qquad z = \begin{matrix} m\{ \\ t\{ \end{matrix} \begin{pmatrix} z^1 \\ z^2 \end{pmatrix}$$

then, the solution of the system $\bar{B}y = z$ can be efficiently calculated using the special partition of the basis $\bar{B}$:

$$y = \left(\begin{array}{c} y^1 \\ \hline y^2 \end{array}\right) = \bar{B}^{-1} \left(\begin{array}{c} z^1 \\ \hline z^2 \end{array}\right) = \begin{pmatrix} B^{-1}\left(z^1 + CQ^{-1}DB^{-1}z^1 - CQ^{-1}z^2\right) \\ Q^{-1}\left(z^2 - DB^{-1}z^1\right) \end{pmatrix} \tag{6}$$

introducing the auxiliary vectors γ_1 and γ_2 the operations stated in (6) can be done in the following way :

Procedure P1:

1) $\gamma_1 \overset{N.E.}{\longleftarrow} B\gamma_1 = z^1$
2) $\gamma_2 = z^1 + CQ^{-1}\left(D\gamma_1 - z^2\right)$
3) $y^1 \overset{N.E.}{\longleftarrow} By^1 = \gamma_1$
4) $y^2 = Q^{-1}\left(z^2 - D\gamma_1\right)$

where $\overset{N.E.}{\longleftarrow}$ denotes a system solved via the spanning tree data structure.

The procedure for the special case $z^1 = (0)$ is of special interest for later developements

Procedure P2:

1) $y^2 = Q^{-1} z^2$
2) $\gamma_1 = -C y^2$
3) $y^1 \overset{N.E.}{\longleftarrow} B y^1 = \gamma_1$

Similarly, the partition of $\bar{B}$ induces a partition on the vectors y and z when solving the system $y'\bar{B} = z'$:

$$y' = (\, y^{1\prime} \;\vert\; y^{2\prime}\,) = (\, z^{1\prime} \;\vert\; z^{2\prime}\,)\,\bar{B}^{-1} =$$
$$= \left(\left(z^{1\prime} + z^{1\prime} B^{-1} C Q^{-1} D - z^{2\prime} Q^{-1} D \right) B^{-1} \quad \left(z^{2\prime} - z^{1\prime} B^{-1} C \right) Q^{-1} \right) \qquad (7)$$

as before, introducing the auxiliary vectors γ_1 and γ_2 the operations stated in (7) can be performed in the four following steps :

Procedure P3:

1) $\gamma_1 \overset{N.E.}{\longleftarrow} \gamma_1' B = z^{1\prime}$
2) $\gamma_2' = z^{1\prime} + \left(\gamma_1' C - z^{2\prime} \right) Q^{-1} D$
3) $y^1 \overset{N.E.}{\longleftarrow} y^{1\prime} B = \gamma_2'$
4) $y^{2\prime} = \left(z^{2\prime} - \gamma_1' C \right) Q^{-1}$

Proceeding in this way the original problem of solving systems $\bar{B}y = z$ and $y'\bar{B} = z'$, each one of dimension $m + t$, has been reduced to solving systems of equations involving matrix B, wich can be computed efficiently exploiting the data structure of the spanning tree , and matrix Q, wich is of dimension t.

2. Computation of a feasible solution

A feasible solution of NNS can be found solving two reduced problems, called here "phase 0 " and "phase 1 ".

2.1 Phase 0.

During phase 0 a "pseudo–feasible " solution $\hat{x}$, that consists of a feasible solution to the NNS problem without side constraints, is found. This is carried out by finding the a feasible initial solution for problem :

$$\begin{array}{rl} \min & cx \\ \text{subject to :} & Ax = r \\ & 0 \leq x \leq u \end{array} \tag{8}$$

with an arbitrary vector of costs coefficients $c \in R^n$.

2.2 Phase 1.

The solution of (8), $\hat{x}$, is feasible for network constraints but, in general, will violate some of the side constraints. Let I denote the set of indeces for the side constrains violated at $\hat{x}$:

$$\mathrm{I} = \left\{ i : T^i \hat{x} > b_i \right\}$$

Inequalities (3) can be transformed into equalities by adding basic slacks in the side constrains not violated at $\hat{x}$ ($f_i > 0, i \notin \mathrm{I}$) and substracting artificial variables (wich represent the infeasibilities) from the remaining ($\equiv$ violated)side constrains ($e_i > 0, i \in \mathrm{I}$). Slacks of violated side constrains are considered non basic ($f_i = 0, i \in \mathrm{I}$) :

$$\left.\begin{array}{ll} T^i \hat{x} + f_i = b_i & i \notin \mathrm{I} \\ T^i \hat{x} - e_i = b_i & i \in \mathrm{I} \end{array}\right\} \qquad T\hat{x} + \hat{\mathbf{I}}\hat{z} = b \tag{9}$$

where $\hat{\mathbf{I}}$ stands for a modified identity matrix defined as :

$$\hat{\mathbf{I}} = \begin{pmatrix} d_1 & & \mathbf{0} \\ & \ddots & \\ \mathbf{0} & & d_t \end{pmatrix} \qquad \text{with} \begin{cases} d_i = +1 & \text{if } i \notin \mathrm{I} \\ d_i = -1 & \text{if } i \in \mathrm{I} \end{cases}$$

and variables $\hat{z}$ of (9) being :

$$\hat{z}_i = \begin{cases} f_i & \text{if } i \notin \mathrm{I} \\ e_i & \text{if } i \in \mathrm{I} \end{cases} \tag{10}$$

Once slacks and artificial variables have been introduced, the following linear problem can be posed :

$$\begin{array}{rll} \min & \hat{\imath}z & \\ \text{subject to :} & Ax & = r \\ & Tx + \hat{\mathbf{I}}z & = b \\ & l \leq x \leq u & z \leq 0 \end{array} \tag{11}$$

The vector of cost coefficients $\hat{\imath}$ is defined as :

$$\hat{\imath} = \begin{cases} \hat{\imath}_j = 0 & \text{if } j \notin \mathrm{I} \\ \hat{\imath}_j = 1 & \text{if } j \in \mathrm{I} \end{cases}$$

that is, the objective function is the sum of infeasibilities, given that $\hat{\imath}z = \sum_{i \in \mathrm{I}} z_i$. A feasible initial point $(\hat{x} \quad | \quad \hat{z})'$ to problem (11) is given by the pseudo–feasible solution $\hat{x}$ and the vector $\hat{z}$ formed as show in (10). Problem (11) is solved through a variation of the Kennington & Helgason algorithm for linear flows with linear side constraints. If the method gets to eliminate all the infeasibilities ($\equiv \hat{\imath}z = 0$), the optimal solution to (11) will be feasible for NNS.

3. Structure of the active set constraint matrix

3.1. Partition of the constraints matrix

Consider the NNS problem introduced in subsection 1.1 with the same assumptions as in subsection 1.3 plus the inclusion of the slacks z for the side constrains and the root arc a :

$$\begin{array}{rll} \min & f(x) & \\ \text{subj. to :} & Ax + e_l a \quad = r & \\ & Tx \quad + \mathbf{I}z = b & \\ & 0 \le x \le u \; ; \; z \ge 0 \; ; \; 0 \le a \le 0 & \end{array} \tag{12}$$

From now on , NNS will denote problem (12).

Consider the constraints matrix of NNS :

$$\bar{A} = \begin{pmatrix} A & e_l & 0 \\ T & 0 & \mathbf{I} \end{pmatrix} \tag{13}$$

partitioned as usual into a *basic matrix* $\bar{B}$, a *superbasic* matrix $\bar{S}$ and a *non basic* matrix $\bar{N}$:

$$\bar{A} = (\bar{B} \quad | \quad \bar{S} \quad | \quad \bar{N}) \tag{14}$$

Let $y = (x \quad | \quad z)'$, $y \in R^{n+t}$, be a feasible point for the NNS . In this notation we consider that the subindex of slack variables z_i denotes the number of the side constrains associated with the slack, so $z_i \equiv \; y_{n+i}$, $i = 1, \ldots, t$.

Consider the usual partition of the variables indices of y induced by (14) :

$$\{\overbrace{1, \ldots, n}^{x}; \overbrace{(n+1), \ldots, (n+t)}^{z}\} = \bar{\mathcal{B}} \cup \bar{\mathcal{S}} \cup \bar{\mathcal{N}} \tag{15}$$

3.2. Internal structure of matrices $\bar{B}$, $\bar{S}$ and $\bar{N}$

It is important to establish the inner structure of the three matrix $\bar{B}$, $\bar{S}$ and $\bar{N}$. We introduced the notation $M^{\mathcal{SUP}}$ to indicate a new matrix formed with the columns of M associated to variables y_i with $i \in \mathcal{SUP}$. Matrix $E^{\mathcal{SUP}}$ denotes a matrix of t rows whose columns are the unitary vectors e_i of R^t associated to the slacks z_i with $(n+i) \in \mathcal{SUP}$.

Matrix $\bar{B}$ in (14) has the same internal partition shown in (5). The basic columns that belong to B are called the *key columns* , and those of C the *nonkey columns* . Extrapolating this internal structure to $\bar{\mathcal{B}}$, two new sets on indices could be defined as

$$\bar{\mathcal{B}} = \mathcal{B} \cup \mathcal{C} \tag{16}$$

and the internal structure of $\bar{B}$ is :

$$\bar{B} = \begin{array}{|c|c|} \hline B & C \\ \hline D & F \\ \hline \end{array} = \begin{array}{|c|c|c|} \multicolumn{1}{c}{m} & \multicolumn{2}{c}{t} \\ \hline A^{\mathcal{B}} & A^{\mathcal{C}} & \mathbf{O} \\ \hline T^{\mathcal{B}} & T^{\mathcal{C}} & E^{\mathcal{C}} \\ \hline \end{array} \begin{array}{l} m \\ t \end{array} \tag{17}$$

Matrices $\bar{S}$ and $\bar{N}$ are similar :

$$\bar{S} = \begin{array}{|c|c|} \multicolumn{1}{c}{\mathrm{s}} & \\ \hline A^{\bar{\mathcal{S}}} & \mathbf{O} \\ \hline T^{\bar{\mathcal{S}}} & E^{\bar{\mathcal{S}}} \\ \hline \end{array} \begin{array}{l} m \\ t \end{array} \tag{18}$$

$$\bar{N} = \begin{array}{|c|c|} \multicolumn{1}{c}{(n-m-s)} & \\ \hline A^{\bar{\mathcal{N}}} & 0 \\ \hline T^{\bar{\mathcal{N}}} & E^{\bar{\mathcal{N}}} \\ \hline \end{array} \begin{array}{l} m \\ t \end{array} \tag{19}$$

And, finally, the matrix of $n+t-s$ active constraints looks like :

$$\hat{A} = \left[\begin{array}{c|cc|cc|c|c} A^{\mathcal{B}} & A^{\mathcal{C}} & \mathbf{O} & A^{\bar{\mathcal{S}}} & \mathbf{O} & A^{\bar{\mathcal{N}}} & \mathbf{O} \\ \hline T^{\mathcal{B}} & T^{\mathcal{C}} & E^{\mathcal{C}} & T^{\bar{\mathcal{S}}} & E^{\bar{\mathcal{S}}} & T^{\bar{\mathcal{N}}} & E^{\bar{\mathcal{N}}} \\ \hline \multicolumn{5}{c|}{\mathbf{O}} & \multicolumn{2}{c}{\mathbf{I}} \end{array}\right] \tag{20}$$

The special structure of these matrices will be exploited in the following section in order to perform efficiently the step of the algorithm proposed. It will be particularly useful in the management of matrix Z, which expands the null–space of the active constraints $\hat{A}$. The null–space matrix Z has been taken as the usual variable reduction matrix :

$$Z = \left(\begin{array}{c} -\bar{B}^{-1}\bar{S} \\ \hline \mathbf{I} \\ \hline 0 \end{array}\right) \tag{21}$$

The treatement of Z in the NNS problem is more difficult than in the pure network flow problem , where $\bar{B} = B$ and $\bar{S} = A^{\mathcal{B}}$. It must be noticed, however , that matrix Z is never computed explicitily in the algorithm developped. The algorithm uses only the internal structure of $\bar{B}$ and $\bar{S}$ shown in (12) and (13) to perform operations with Z and Z'

4. The algorithm.

In order to define the NNS problem, some information must be given to the algorithm at the beginning of the optimization process, and kept accessible during the whole process. This information is :

1) The objective function $f(x)$ and its gradient $g(x) = \nabla f(x)$. The Hessian is not really needed.
2) The vectors containing the description of the network.
3) The (probably sparse) representation of the side constrains matrix T.
4) The vector of arc capacities u.

Having this information stored in core, suppose that our algorithm has guided the optimization process to the vector $y = (y^{\bar{\mathcal{B}}} \mid y^{\bar{\mathcal{S}}} \mid y^{\bar{\mathcal{N}}})$, feasible for NNS. In such a point we assume the next items to be available :

5) Some internal representation of the set of indices $\bar{\mathcal{B}}$,$\bar{\mathcal{S}}$ and $\bar{\mathcal{N}}$.
6) The vectors describing the spanning tree .
7) The inverse of the working basis , stored in some efficient way.
8) Depending on the specific implementation of the algorithm, it would be necessary to keep in memory some approximation to the *reduced Hessian* .

4.1. Computation of the Lagrange multipliers

Lagrange multipliers are defined as the solution to the system arisinig from the first order necessary condition of minimum :

$$\hat{A}'\lambda = g \qquad ; \qquad \begin{pmatrix} \bar{B}' & 0 \\ \bar{S}' & 0 \\ \bar{N}' & \mathbf{I} \end{pmatrix} \begin{pmatrix} \pi \\ \hline \sigma \end{pmatrix} = \begin{pmatrix} g^{\bar{\mathcal{B}}} \\ \hline g^{\bar{\mathcal{S}}} \\ \hline g^{\bar{\mathcal{N}}} \end{pmatrix} \tag{22}$$

Vector π is intensivily used during each iteration, whereas σ is only needed when the active set of constraints change. Both can be easily computed using the structure of $\hat{A}$ studied before. The Lagrange multipliers π are calculated from the system :

$$\pi' = g^{\bar{\mathcal{B}}\prime}\bar{B}^{-1} \qquad ; \qquad (\pi^{1\prime} \mid \pi^{2\prime}) = (g^{\mathcal{B}\prime} \mid g^{\mathcal{C}\prime})\,\bar{B}^{-1} \tag{23}$$

which can be solve efficiently via procedure P3. π^1 represents the Lagrange multipliers associated with the network constraints and π^2 is linked to the side constrains .

The expression of the Lagrange multipliers σ taking into account the partition of $g^{\bar{\mathcal{N}}}$, $\bar{N}$ and π is :

$$\sigma = \begin{pmatrix} \sigma^1 \\ \hline \sigma^2 \end{pmatrix} = \begin{pmatrix} g^{\bar{\mathcal{N}}_x} \\ \hline 0 \end{pmatrix} - \begin{pmatrix} A^{\bar{\mathcal{N}}\prime} & T^{\bar{\mathcal{N}}\prime} \\ 0 & E^{\bar{\mathcal{N}}\prime} \end{pmatrix} \begin{pmatrix} \pi^1 \\ \hline \pi^2 \end{pmatrix} =$$

$$= \begin{pmatrix} g^{\bar{\mathcal{N}}_x} \\ \hline 0 \end{pmatrix} - \begin{pmatrix} A^{\bar{\mathcal{N}}\prime}\pi^1 + T^{\bar{\mathcal{N}}\prime}\pi^2 \\ E^{\bar{\mathcal{N}}\prime}\pi^2 \end{pmatrix} \tag{24}$$

where $g^{\bar{\mathcal{N}}_x}$ stands for the components of $g^{\bar{\mathcal{N}}}$ related with the true variables x.

The computational procedure used to evaluate σ is :

Procedure P4:

1) For $\forall i \in \bar{\mathcal{S}}$ do :
   1.1) If $i > n$ (slacks) : $\sigma_i := -\pi_i$
   1.2) If $i \leq n$ (arcs) :
      1.2.1) Find "from node" k and "to node" l for arc i.
      1.2.2) $\gamma_1 := \pi^1_k - \pi^2_l$
      1.2.3) $\gamma_2 := T^{\bar{\mathcal{N}}\prime}{}_i \pi^2$
      1.2.4) $\sigma_i := g^{\bar{\mathcal{N}}}_i - \gamma_1 - \gamma_2$

Notice that only one matrix-vector product $T^{\bar{\mathcal{N}}\prime}\pi^2$ of reduced dimension

$$(\; n \; - \; m \; - \; s \;) \; \times \; t$$

must be computed explicitily in step 1.2.3) .

4.2. Computation of the reduced gradient

Once π has been found, the *reduced gradient* $g_z = Z'g$; $g_z \in R^s$ can be obtained from the expression :

$$g_z = \left(-\left(\bar{B}^{-1}\bar{S}\right)' \;\middle|\; \mathbf{I} \;\middle|\; 0 \right) \begin{pmatrix} g^{\bar{\mathcal{B}}} \\ \hline g^{\bar{\mathcal{S}}} \\ \hline g^{\bar{\mathcal{N}}} \end{pmatrix} = \begin{pmatrix} g_x^{\bar{\mathcal{S}}} \\ \hline 0 \end{pmatrix} - \begin{pmatrix} A^{\bar{\mathcal{S}}\prime}\pi^1 + T^{\bar{\mathcal{S}}\prime}\pi^1 \\ E^{\bar{\mathcal{S}}\prime}\pi^2 \end{pmatrix} \tag{25}$$

It must be noticed that σ and g_z are found via the same procedure.

Once g_z is computed, the next step is to find a direction of search along which the objective function decreases and feasibility is preserved.

4.3. Search directions

Supose that vector p_z is an aproximation to the solution of the unconstrained problem :

$$\min \frac{1}{2} p_z' Z' H Z p_z + g' Z p_z \tag{26}$$

which represents the minimization of a quadratic local aproximation to the projection of function $f(x)$ over the null–space of $\hat{A}$. Matrix $H_z = Z'HZ$, $H_z \in R^{s \times s}$ is the

reduced Hessian and vector $g_z = Z'g$, $g_z \in R^s$ is the *reduced gradient* . p_z is obtained solving the system :

$$Z'HZp_z = -Z'g \tag{27}$$

Once p_z has been found, a feasible direction of movement $p = (p^{\bar{\mathcal{B}}} \mid p^{\bar{\mathcal{S}}} \mid p^{\bar{\mathcal{N}}})$ can be obtained from the relation :

$$p = Zp_z = \begin{pmatrix} -\bar{B}^{-1}\bar{S}p_z \\ \hline p_z \\ \hline 0 \end{pmatrix} \tag{28}$$

The algorithmic procedure to compute (28) is :

Procedure P5:

1) $p^{\bar{\mathcal{N}}} = (0)$
2) $p^{\bar{\mathcal{S}}} = p_z$
3) For $\forall i \in \bar{\mathcal{S}}$ do :

3.1) If $i \leq n$ (arcs) : $\gamma \xleftarrow{\text{P1}} \bar{B} \begin{pmatrix} \gamma^1 \\ \hline \gamma^2 \end{pmatrix} = \begin{pmatrix} A_i \\ \hline T_i \end{pmatrix}$

If $i > n$ (slacks) : $\gamma \xleftarrow{\text{P2}} \bar{B} \begin{pmatrix} \gamma^1 \\ \hline \gamma^2 \end{pmatrix} = \begin{pmatrix} 0 \\ \hline e_{i-n} \end{pmatrix}$

3.2) $\begin{pmatrix} p^{\mathcal{B}} \\ \hline p^{\mathcal{C}} \end{pmatrix} := \begin{pmatrix} p^{\mathcal{B}} \\ \hline p^{\mathcal{C}} \end{pmatrix} - p_{zi} \begin{pmatrix} \gamma^1 \\ \hline \gamma^2 \end{pmatrix}$

Two different alternative techniques for solving system (27) have been implemented :

1) A truncated Newton method (TNM).

2) A quasi–Newton method (QNM).

The truncated–Newton method follows the strategy exposed in [4]. It is based on the solution of system (27) by a *conjugated gradient* (CG) method.

With regards to the use of the second derivatives, the CG algorithm only needs the aproximation to the Hessian H to compute the products $Z'HZd$, where the vectors d are the directions generated by the CG method in each iteration. This products are computed actually by the procedure :

Procedure P6:

1) $\gamma_1 \overset{\text{P5}}{\longleftarrow} Zd$
2) $\gamma_2 = \dfrac{g(x+\epsilon\gamma_1) - g(x)}{\epsilon} \approx H\gamma_1$
3) $\gamma_2 \overset{\text{P4}}{\longleftarrow} Z'\gamma_1$

in step 2) of P6 a forward finite difference technique is applied to approximate the product $H\gamma_1$. Proceeding in this way it is neither necessary to compute, nor to store, any approximation to Hessian matrix.

The quasi–Newton version follows the metodology exposed in [7]. The Cholesky factors R of an approximation to the reduced Hessian ($R'R \approx H_z$) must be stored, updated and retriangularized whenever a change in the matrix H_z occurs. Matrix H_z is modified when :

1) $\bar{\mathcal{B}}$ or $\bar{\mathcal{S}}$ change (an exchange between $\mathcal{B}$ and $\mathcal{C}$ doesn't requires R to be update).
2) A nonzero step is performed in the real variables.

In the first case, after the addition or removal of certain number of rows/columns of R, the factor R is retriangularized via succesive Givens rotations. When a change in the variables is made , factors $R'R$ are updated with a complementary DFP formula.

The two alternatives techniques implemented have the common features of :

1) Only an approximation to the solution of (27) is found.
2) No second derivatives are used.

and differ in that :

1) *Memory requirements :* TNM has not extra memory requirements, while QNM must keep stored the factors of the aproximation to the reduced Hessian.

2) *Extra gradient evaluations :* QNM only needs one extra evaluation of $g(x)$ for the updating of the reduced Hessian aproximation when the active set is change. On the contrary, TNM needs, in the implementation developped, one evaluation per CG–iteration (P6 , step 2)). So, if problem NNS has a costly objective function, QNM should be preferable to TNM.

3) *Computational effort :* Leaving aside the evaluation of $g(x)$, TNM needs one call of procedures P3, P4 and P5 at each CG–iteration. QNM must update and retriangularize the upper matrix R.

4.4. Linesearch

The solution to (27) provides a descent direction that :

1) Preserves feasibility for the set of active constraints $\forall \alpha > 0$
2) Violates the simple bound of a basic or superbasic variable for certain $\alpha > \bar{\alpha} > 0$

Therefore, if $\alpha^{\bar{\mathcal{B}}}$ and $\alpha^{\bar{\mathcal{S}}}$ denote the maximal step length allowed by variables in $\bar{\mathcal{B}}$ and $\bar{\mathcal{S}}$, respectively, a limited linesearch with $0 \leq \alpha \leq \bar{\alpha} = \min\{\alpha^{\bar{\mathcal{B}}}, \alpha^{\bar{\mathcal{S}}}\}$ must be executed in order to the superbasic and basic bounds not to be violated by the iterated point. In fact, we must solve a nonlinear problem with simple constraints.

Bertsekas in [1] introduces a method to solve this kind of problems where more than one variable can be set to one of its bound simultaneously. This is a very useful property for NNS problem, because it introduces the posibility of eliminating more than one superbasic variable in each reduced gradient iteration.

We have follow the scheeme presented by Toint & Tuyttens in [8] for this special linesearch. We will refer to this linesearch as LSBE. In particular, the actual algorithm makes use of the *quasi–active bounds* strategy for finding the search direction p. In this strategy the algorithm for finding the descent direction p acts only over the components of p associated with superbasic variables without quasi–active bounds, and takes for the other components the steepest descent direction.

When a Bertsekas step (that means, $\alpha^* > \bar{\alpha} \equiv \alpha^{\bar{\mathcal{S}}}$) cannot be performed, a cubic fit is made between $\alpha = 0$ and $\alpha = \bar{\alpha}$ using $f(x)$,$g(x)$,$f(x + \bar{\alpha}p)$ and $g(x + \bar{\alpha}p)$. If the cubic fit fails, line search of the type $\alpha_{k+1} = \beta^j \alpha_k$, $j = 0, 1, \ldots$; $0 \leq \beta \leq 1$ is used. A cubic fit is also used when $y^{\bar{\mathcal{S}}} \equiv z^{\bar{\mathcal{S}}}$, that is, the only superbasic variable existing is a slack, and when $\bar{\mathcal{S}} = \emptyset$. We will refer to this linesearch as LSAC.

Both subroutines, LSBE and LSAC provide, not only and estimation of α^*, but also the new values of y, $f(x)$ and $g(x)$.

5. Update of Q^{-1} and R

Matrices Q^{-1} and R are stored, updated and, eventually, recomputed, during the optimization process. We deal in this section with those processes, commenting only the most peculiar characteristics of the algorithm implemented.

5.1. Working basis inverse update

Kennington & Helgason present in [6] the way to update matrix Q^{-1} after a simplex pivot in the linear network case. Here are presented only the results. For a more detailed explanation, see [6] pag. 172–174.

Consider that, after a pivot, the relation between the old and the new basic matrix is :

$$\bar{B}_{i+1}^{-1} = E\bar{B}_i^{-1} \tag{29}$$

with :

$$E = \begin{array}{c} \\ m \\ t \end{array} \begin{array}{c} \overbrace{}^{m} \quad \overbrace{}^{t} \\ \left(\begin{array}{cc} E_1 & E_2 \\ E_3 & E_4 \end{array} \right) \end{array} \tag{30}$$

where E (eta) may be either an elementary column matrix or a permutation matrix.

The following cases can be distinguish :

1) The nonkey column $p \in \mathcal{C}$ leaves the basis.

1.1) If the entering variable q is to be placed in the same position left by p :

$$Q_{i+1}^{-1} = E_4 Q_i^{-1} \tag{UIQ1}$$

being E the elementary column matrix that updates $\bar{B}_i^{-1}$.

1.2) If the entering variable q is to be placed in the position of variable $k \neq p$:

$$Q_{i+1}^{-1} = Q_i^{-1} P_{pk} \tag{UIQ2}$$

being P_{pk} an elementary permutation matrix. Go to case *1.1)*

2) The key column $p \in \bar{\mathcal{B}}$ leaves the basis.

2.1) If there is an arc $q \in \mathcal{C}$ forming cycle with the leaving arc p, exchange columns p and q of $\bar{B}$:

$$Q_{i+1}^{-1} = E_5 Q_i^{-1} \tag{UIQ3}$$

being E_5 the elementary row matrix :

$$E_5 = \begin{pmatrix} \mathbf{I} & 0 & 0 \\ \cdots\cdots\cdots & & \\ & -e_p' B^{-1} C & \\ \cdots\cdots\cdots & & \\ 0 & 0 & \mathbf{I} \end{pmatrix}$$

and go to case *1)*.

2.2) If doesn't exists any arc $q \in \mathcal{C}$ forming cicle with the leaving arc p :

$$Q_{i+1}^{-1} = Q_i^{-1} \tag{UIQ4}$$

Kennington & Helgason didn't consider case 1 divided into subcases but the algorithm presented needs the new update UIQ2 because it usually change the order of two nonkey columns when the entering variable q is a slack.

The inverse of the working basis is found once at the beginning of the first iteration and updated when a change in the set $\bar{\mathcal{B}}$ occurs. Reinversion takes place after a certain number of updatings. The Q^{-1} is stored as an eta file with a set of complementary vectors. This eta file holds the coefficients of the eta columns of the last reinversion and the information of all the updatings made since the last reinversion. This information is either coefficients of column/row eta (UIQ1 and UIQ3) or two numbers denoting a nonkey column permutation (UIQ2). The reinversion routines implements the row and column reordering system proposed by Hellerman & Rarick in [5] and partial pivoting to ensure numerical stability.

5.2. Updating the approximation to the factors of the Hessian

The management of factors R follows the procedure exposed by Murtagh & Saunders in [7]. We only comment some peculiarities related to the special NNS problem.

Four differents updates of factor R can take place :

1) *UR1 : Addition of one superbasic variable.* The update formula is :

$$\tilde{R}'\tilde{R} = \begin{pmatrix} R'R & Z'v \\ v'Z & z'v \end{pmatrix}$$

In order to update R it is necessary to perform a set of matrix operations that can be computed efficiently using the procedures exposed in early sections. Those matrix operations and its related procedures are :

a) $z = \bar{B}^{-1}\bar{N}_q$ found via P1, if $q \leq n$ (arc) and via P2 if $q > n$ (slack).

b) Aproximation of vector $v = Hz$ by finite differences. Equivalent to step 2) of procedure P6.

c) $r = Z'v$ found using procedure P4.

2) *UR2 : Quasi–Newton updates.* Takes into account a change in the superbasic variables. The complementary DFP formula has been used.

3) *UR3 : Change of matrix $\bar{B}$.* In this case, the basic variable $p \in \bar{\mathcal{B}}$ is exchange with the superbasic variable $q \in \bar{\mathcal{S}}$. The modification of R is :

$$\tilde{R}'\tilde{R} = (\mathbf{I} + ve_q')R'R(\mathbf{I} + e_q v')$$

In order to find vector v two matrix operations must be done :

a) $\bar{\beta}^{p\prime} = e_p' \bar{B}^{-1}$. This operation is computed by P3. If $p \in \mathcal{B}$, P3 ,must be applied setting $z^1 = e_p$ and $z^2 = (0)$ otherwise, if $p \in \mathcal{C}$, the correct assignment is $z^1 = (0)$ and $z^2 = e_p$.

b) $w = \bar{S}' \bar{\beta}^p = \begin{pmatrix} A^{\bar{\mathcal{S}}\prime} \bar{\beta}^{p1} + T^{\bar{\mathcal{S}}\prime} \bar{\beta}^{p2} \\ E^{\bar{\mathcal{S}}\prime} \bar{\beta}^{p2} \end{pmatrix}$. Vector w can be evaluated with a procedure similar to P4.

It must be stressed that the update of g_z and π, wich is made using vectors $\bar{\beta}^p$ and w, takes also advantage of this special computation.

3) *UR4 : Removal of a superbasic variable* $q \in \bar{\mathcal{S}}$. The same update than in the general case.

6. Statement of the algorithm

The structure of the proposed algorithm is outlined in this section. We consider that phase 0 and phase 1 have been performed, providing a feasible vector y. The sets $\mathcal{C}$ and $\bar{\mathcal{S}}$ with a subindex x or z denote the subset of indices associated with arcs or slacks.

[0] Let $y = (y^{\bar{\mathcal{B}}} \mid y^{\bar{\mathcal{N}}})$ (initially $\bar{\mathcal{S}} = \emptyset$).

[0.1] Compute $f(x)$ and $g(x)$.

[0.2] Compute $\left(\begin{array}{c} \pi^1 \\ \hline \pi^2 \end{array} \right)' \xleftarrow{\text{P3}} g^{\bar{\mathcal{B}}\prime} \bar{B}^{-1}$.

[0.3] Compute Q_0^{-1}.

[1] If $\|g_z\| \geq$ TGR goto [3].

[2] *Change of the active set.*

[2.1] Compute σ^2 through $\sigma^2 = -E^{\bar{\mathcal{N}}\prime} \pi^2$.
If the slack y_q $(\equiv z_{q-n})$ can leave $\bar{\mathcal{N}}$, goto [2.4]

[2.2] Compute σ^1 through procedure P4.
If the arc y_q can leave $\bar{\mathcal{N}}$, goto [2.4]

[2.3] Go to [11]

[2.4] *Updates*
$\bar{\mathcal{N}} := \bar{\mathcal{N}} \backslash q$.

If $q \leq n$, $\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \cup q$.
If $q > n$, $\bar{\mathcal{S}}_z := \bar{\mathcal{S}}_z \cup q$.
Update π and g_z.
If QNM used, update UR1.

[3] *Computing a feasible descent direction.*

[3.1] Find $\bar{\mathcal{S}}^{\mathcal{QA}}$ the set of superbasic variables with quasi–active bounds.
[3.2] Set $p_z^{\mathcal{QA}} := -g_z^{\mathcal{QA}}$.
[3.3] Find p_{z_q} , $q \notin \bar{\mathcal{S}}^{\mathcal{QA}}$ through TNM or QNM.
[3.4] Find $p \xleftarrow{\text{P5}} p = Zp_z$.

[4] *Linesearch*

[4.1] Find $\alpha^{\bar{\mathcal{B}}}$ and $\alpha^{\bar{\mathcal{S}}}$, the maximal step lenght for basic and superbasic variables. If $\bar{\alpha} = \min\{\alpha^{\bar{\mathcal{B}}}, \alpha^{\bar{\mathcal{S}}}\} = 0$, set $\alpha^* = \bar{\alpha}$. Goto [6].
[4.2] If $\bar{\mathcal{S}} \equiv \bar{\mathcal{S}}_z$, find α^*,$\tilde{y} = y + \alpha^* p$, $f(\tilde{y})$ and $g(\tilde{y})$ via LSAC.
If $\bar{\mathcal{S}} \not\equiv \bar{\mathcal{S}}_z$, find α^*,$\tilde{y} = y + \alpha^* p$, $f(\tilde{y})$ and $g(\tilde{y})$ via LSBE.
[4.3] *Updates for the variables change*
Compute $g_z \xleftarrow{\text{P4}} Z'g$.
Compute $\pi \xleftarrow{\text{P3}} g^{\bar{\mathcal{B}}\prime}\bar{B}^{-1}$.
If QNM used, perform update UR2.

[5] If $\alpha^* < \bar{\alpha}$, goto [8]

[6] $\alpha^* = \alpha^{\bar{\mathcal{B}}}$ *: basis change, variable* p *leave* $\bar{B}$

[6.1] If $\bar{\mathcal{S}}_z \neq \emptyset$, select, if possible, slack q to become basic. Otherwise, select a suitable superbasic arc $q \in \bar{\mathcal{S}}_x$ to become basic
[6.2] If $\bar{\alpha} = \alpha^{\mathcal{B}}$:

[6.2.1] If $q \in \bar{\mathcal{S}}_z$ then :

Select arc $k \in \mathcal{C}_x$ to interchange with $p \in \mathcal{B}$

$\mathcal{B} := \mathcal{B}\backslash p \cup k$
$\mathcal{C} := \mathcal{C}\backslash k \cup p$
Update Q^{-1} using UIQ3.
Interchange $q \in \bar{\mathcal{S}}_z$ with $p \in \mathcal{C}$

$\bar{\mathcal{S}}_z := \emptyset$,$\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \cup p$
$\mathcal{C} := \mathcal{C}\backslash p \cup q$
If internal reordering is needed, do UIQ2 and UIQ1
If internal reordering isn't needed, UIQ1
Goto [7]

[6.2.2] If $q \in \bar{\mathcal{S}}_x$ then :

If p forms cycle with q

If $\exists k \in \mathcal{C}_x$ forming cycle with p, do :

$\mathcal{B} := \mathcal{B} \backslash p \cup k$
$\mathcal{C} := \mathcal{C} \backslash k \cup p$
Update Q^{-1} using UIQ3.
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$
Update Q^{-1} using UIQ1.
Goto [7]

If $\nexists k \in \mathcal{C}_x$ forming cycle with p, do:

$\mathcal{B} := \mathcal{B} \backslash p \cup q$
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$
Update Q^{-1} using UIQ5.
Goto [7]

If p doesn't forms cycle with q

Select arc $k \in \mathcal{C}_x$ to interchange with $p \in \mathcal{B}$ do:

$\mathcal{B} := \mathcal{B} \backslash p \cup k$
$\mathcal{C} := \mathcal{C} \backslash k \cup p$
Update Q^{-1} using UIQ3.
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$
Update Q^{-1} using UIQ1.
Goto [7]

[6.3] If $\bar{\alpha} = \alpha^{\mathcal{C}_x}$:

[6.3.1] If $q \in \bar{\mathcal{S}}_z$ then :

Interchange $q \in \bar{\mathcal{S}}_z$ with $p \in \mathcal{C}$

$\bar{\mathcal{S}}_z := \emptyset, \bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \cup p$
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
If internal reordering is needed, do UIQ2 and UIQ1
If internal reordering isn't needed, UIQ1
Goto [7]

[6.3.2] If $q \in \bar{\mathcal{S}}_x$ then :

Interchange $q \in \bar{\mathcal{S}}_x$ with $p \in \mathcal{C}_x$

$\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \backslash q \cup p$
$\mathcal{C}_x := \mathcal{C}_x \backslash p \cup p$
Update Q^{-1} using UIQ1.
Goto [7]

[6.4] If $\bar{\alpha} = \alpha^{\mathcal{C}_z}$:

[6.4.1] If $q \in \bar{\mathcal{S}}_z$ then :

Interchange $q \in \bar{\mathcal{S}}_z$ with $p \in \mathcal{C}_z$

$\bar{\mathcal{S}}_z := \bar{\mathcal{S}}_z \backslash q \cup p$
$\mathcal{C}_z := \mathcal{C}_z \backslash p \cup q$
Update Q^{-1} using UIQ1.
Goto [7].

[6.4.2] If $q \in \bar{\mathcal{S}}_x$ then :

Interchange $q \in \bar{\mathcal{S}}_x$ with $p \in \mathcal{C}_z$

$\bar{\mathcal{S}}_z := \bar{\mathcal{S}}_z \cup q$
$\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \backslash q$
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
If internal reordering is needed, do UIQ2 and UIQ1
If internal reordering isn't needed, UIQ1
Goto [7].

[7] *Update* $\bar{\mathcal{S}}$

[7.1] Remove from $\bar{\mathcal{S}}$ all the superbasic variables set to one of its bounds by LSBE.
[7.2] If QNM used, apply UR4 as many times as superbasic variables deleted.

[8] If $\bar{\mathcal{S}}_z \neq \emptyset$ do :

[8.1] Select a suitable $p \in \mathcal{C}$ to interchange with q.
[8.2] Updates
$\mathcal{C} := \mathcal{C} \backslash p \cup q$.
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$.
g_z,π as usual, using a*)* and *b)* of UR3.
If QNM used, perform UR3.

[9] *Reinversion of* Q Performed each NRI updatings of Q^{-1}
[10] Goto [1]
[11] STOP : optimal solution found.

this is the basic structure of the algorithm developed. However, some remarks must be made.

6.3. Influence of a basic columns reordering on UR3

At steps [6.2.1] , [6.3.1] and [6.4.2] a two–column permutation can be applied. It has been explained above how this internal interchange affects the update of the working basis. It can be proved that a basic column permutation doesn't affect the standard update UR3, so, no special update of R must be performed in this cases.

6.4. Management of the slacks

The algorithm described dispenses an special treatement to the slacks. First, in step [2], when a nonbasic variable to leave its active bound is being looked for, slacks

are scanned before than arcs. If a slack is found to be a good candidate, then it becomes superbasic, but only temporarily, because it is interchanged with a nonkey arc at the end of the present iteration (step [8]). This strategy tries to improve the stability of the solution of the systems of equations with coefficient matrix Q introducing in it as many identity columns as possible.

6.5. Trial step $p_z = \pm e_q$

There is an alternative way to manage the variable q which has been relaxed from one of its bounds in step [2] and added to $\bar{\mathcal{S}}$. It consists in fixing the values of the remaining superbasic variables and performing a null–space step of the form :

$$p_z = \begin{pmatrix} p_{z_1} \\ \vdots \\ p_{z_s} \\ \hline p_{z_q} \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \hline \pm 1 \end{pmatrix} \tag{31}$$

with $p_{z_q} = +1$ if the new superbasic variable has been relaxed from its lower bound and $p_{z_q} = -1$ otherwise. It can be proved easily that this step is a descent direction. The new component of the reduced gradient is equal to the Lagrange multiplier associated with variable q :

$$g_{z_q} = -\sigma_q \tag{32}$$

testing the descent condition :

$$g'_z p_z = -\sigma_q p_{z_q} < 0 \tag{33}$$

the last inequality holds if we have selected q properly.

Once computed $p^{\bar{\mathcal{B}}}$ from p_z a linesearch LSCF is made. If the result of linesearch is the maximal step length, then superbasic variable $s+1$ is removed from $\bar{\mathcal{S}}$ and added to $\bar{\mathcal{B}}$, so, the number of superbasic variables has not increased. If such a situation happens, the iteration performed is nothing different than a simplex pivot between superbasic $s+1$ and the basic variable associated with the maximal step length.

If linesearch provides an $\alpha^* < \bar{\alpha}$, then if the temporary superbasic variable is an arc, it remains in $\bar{\mathcal{S}}$. If it is a slack, then an interchange with a nonkey arc is done as in step [8].

6.6. Description of the pivot operations in terms of the network structure

The simultaneous presence in the network of arcs $\mathcal{C}_x$ and arcs $\bar{\mathcal{S}}_x$ makes the interpretation and the management of the pivot operation a bit more difficult than in the pure nonlinear network flow problems. In the pure nonlinear network problem, it can be assured that if the flow of a basic arc p change due to a step in the superbasic arcs then, it exist at least one superbasic arc that forms cycle with arc p. However, in the presence of side constraints, it is possible for an arc $p \in \mathcal{B}$ to change its value without

forming cycle with any superbasic arc $q \in \bar{\mathcal{S}}_x$. From (6) and (28), the step $p^{\mathcal{B}}$ can be expressed as :

$$p^{\mathcal{B}} = -B^{-1}\,(\,A^{\bar{\mathcal{S}}} \;\;|\;\; 0\,)\,p_z - B^{-1}\,(\,A^{\mathcal{C}} \;\;|\;\; 0\,)\left(Q^{-1}DB^{-1}\,(\,A^{\bar{\mathcal{S}}} \;\;|\;\; 0\,) - Q^{-1}T^{\bar{\mathcal{S}}}\right)p_z \tag{34}$$

From (34) it is clear that :

$$p_p^{\mathcal{B}} \neq 0 \Rightarrow \beta^p A^{\bar{\mathcal{S}}} \neq (0) \quad \textit{or} \quad \beta^p A^{\mathcal{C}} \neq (0)$$

where β^p is the row of B^{-1} associated with arc p.

The nonkey arcs can also change its value without been in a superbasic cycle, as shown in the following expression, build also from (6) and (28) :

$$p^{\mathcal{C}} = -Q^{-1}\left((\,T^{\bar{\mathcal{S}}} \;\;|\;\; 0\,) - DB^{-1}\,(\,A^{\bar{\mathcal{S}}} \;\;|\;\; 0\,)\right)p_z \tag{35}$$

If a superbasic slack q is selected in [6.1] to become basic, and the variable p is an arc from the spanning tree , then, it will be necessary to perform a key non–key column interchange before the slack q has been entered in $\bar{\mathcal{B}}$. It can be assured that this previous interchange will be always possible because of the expression of the pivot element associated with slack q. If variable q has been selected to become basic, its pivot element must be different from zero. The pivot vector associated with slack q is :

$$w = \bar{B}^{-1}\bar{S}_q = \bar{B}^{-1}\begin{pmatrix} 0 \\ \hline e_q \end{pmatrix} \tag{36}$$

and the component associated with arc p is :

$$w_p = -\beta^p C Q^{-1} e_q \tag{37}$$

So,

$$w_p \neq 0 \Rightarrow \beta^p C \neq 0$$

, that is, there exists nonkey arcs forming cycle with arc p.

7. Statement of the algorithm

The structure of the proposed algorithm is outlined in this section. We consider that phase 0 and phase 1 have been performed, providing a feasible vector y. The sets $\mathcal{C}$ and $\bar{\mathcal{S}}$ with a subindex x or z denote the subset of indices associated with arcs or slacks.

[0] Let $y = (\,y^{\bar{\mathcal{B}}} \;\;|\;\; y^{\bar{\mathcal{N}}}\,)$ (initially $\bar{\mathcal{S}} = \emptyset$).

[0.1] Compute $f(x)$ and $g(x)$.

[0.2] Compute $\begin{pmatrix} \pi^1 \\ \hline \pi^2 \end{pmatrix}' \xleftarrow{\text{P3}} g^{\bar{\mathcal{B}}\prime}\bar{B}^{-1}$.

[0.3] Compute Q_0^{-1}.

[1] If $\|g_z\| \geq$ TGR goto [3].

[2] *Change of the active set.*

[2.1] Compute σ^2 through $\sigma^2 = -E^{\bar{\mathcal{N}}\prime}\pi^2$.
If the slack y_q ($\equiv z_{q-n}$) can leave $\bar{\mathcal{N}}$, goto [2.4]
[2.2] Compute σ^1 through procedure P4.
If the arc y_q can leave $\bar{\mathcal{N}}$, goto [2.4]
[2.3] Go to [11]
[2.4] *Updates*
$\bar{\mathcal{N}} := \bar{\mathcal{N}} \backslash q$.
If $q \leq n$, $\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \cup q$.
If $q > n$, $\bar{\mathcal{S}}_z := \bar{\mathcal{S}}_z \cup q$.
Update π and g_z.
If QNM used, update UR1.

[3] *Computing a feasible descent direction.*

[3.1] Find $\bar{\mathcal{S}}^{\mathcal{QA}}$ the set of superbasic variables with quasi–active bounds.
[3.2] Set $p_z^{\mathcal{QA}} := -g_z^{\mathcal{QA}}$.
[3.3] Find p_{z_q} , $q \notin \bar{\mathcal{S}}^{\mathcal{QA}}$ through TNM or QNM.
[3.4] Find $p \xleftarrow{\text{P5}} p = Zp_z$.

[4] *Linesearch*

[4.1] Find $\alpha^{\bar{\mathcal{B}}}$ and $\alpha^{\bar{\mathcal{S}}}$, the maximal step lenght for basic and superbasic variables. If $\bar{\alpha} = \min\{\alpha^{\bar{\mathcal{B}}}, \alpha^{\bar{\mathcal{S}}}\} = 0$, set $\alpha^* = \bar{\alpha}$. Goto [6].
[4.2] If $\bar{\mathcal{S}} \equiv \bar{\mathcal{S}}_z$, find α^*,$\tilde{y} = y + \alpha^* p$, $f(\tilde{y})$ and $g(\tilde{y})$ via LSAC.
If $\bar{\mathcal{S}} \not\equiv \bar{\mathcal{S}}_z$, find α^*,$\tilde{y} = y + \alpha^* p$, $f(\tilde{y})$ and $g(\tilde{y})$ via LSBE.
[4.3] *Updates for the variables change*
Compute $g_z \xleftarrow{\text{P4}} Z'g$.
Compute $\pi \xleftarrow{\text{P3}} g^{\bar{\mathcal{B}}\prime}\bar{B}^{-1}$.
If QNM used, perform update UR2.

[5] If $\alpha^* < \bar{\alpha}$, goto [8]

[6] $\alpha^* = \alpha^{\bar{\mathcal{B}}}$ *: basis change, variable* p *leave* $\bar{B}$

[6.1] If $\bar{\mathcal{S}}_z \neq \emptyset$, select, if possible, slack q to become basic. Otherwise, select a suitable superbasic arc $q \in \bar{\mathcal{S}}_x$ to become basic
[6.2] If $\bar{\alpha} = \alpha^{\mathcal{B}}$:

[6.2.1] If $q \in \bar{\mathcal{S}}_z$ then :

Select arc $k \in \mathcal{C}_x$ to interchange with $p \in \mathcal{B}$

$\mathcal{B} := \mathcal{B} \backslash p \cup k$
$\mathcal{C} := \mathcal{C} \backslash k \cup p$
Update Q^{-1} using UIQ3.

Interchange $q \in \bar{\mathcal{S}}_z$ with $p \in \mathcal{C}$

$\bar{\mathcal{S}}_z := \emptyset, \bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \cup p$
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
If internal reordering is needed, do UIQ2 and UIQ1
If internal reordering isn't needed, UIQ1

Goto [7]

[6.2.2] If $q \in \bar{\mathcal{S}}_x$ then :

If $\exists k \in \mathcal{C}_x$ forming cycle with p, do :

$\mathcal{B} := \mathcal{B} \backslash p \cup k$
$\mathcal{C} := \mathcal{C} \backslash k \cup p$
Update Q^{-1} using UIQ3.
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$
Update Q^{-1} using UIQ1.
Goto [7]

If $\nexists k \in \mathcal{C}_x$ forming cycle with p, do:

$\mathcal{B} := \mathcal{B} \backslash p \cup q$
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$
Update Q^{-1} using UIQ5.
Goto [7]

[6.3] If $\bar{\alpha} = \alpha^{\mathcal{C}_x}$:

[6.3.1] If $q \in \bar{\mathcal{S}}_z$ then :

Interchange $q \in \bar{\mathcal{S}}_z$ with $p \in \mathcal{C}$

$\bar{\mathcal{S}}_z := \emptyset, \bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \cup p$
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
If internal reordering is needed, do UIQ2 and UIQ1
If internal reordering isn't needed, UIQ1

Goto [7]

[6.3.2] If $q \in \bar{\mathcal{S}}_x$ then :

Interchange $q \in \bar{\mathcal{S}}_x$ with $p \in \mathcal{C}_x$

$\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \backslash q \cup p$
$\mathcal{C}_x := \mathcal{C}_x \backslash p \cup p$
Update Q^{-1} using UIQ1.

Goto [7]

[6.4] If $\bar{\alpha} = \alpha^{\mathcal{C}_z}$:

[6.4.1] If $q \in \bar{\mathcal{S}}_z$ then :

Interchange $q \in \bar{\mathcal{S}}_z$ with $p \in \mathcal{C}_z$

$\bar{\mathcal{S}}_z := \bar{\mathcal{S}}_z \backslash q \cup p$
$\mathcal{C}_z := \mathcal{C}_z \backslash p \cup q$
Update Q^{-1} using UIQ1.
Goto [7].

[6.4.2] If $q \in \bar{\mathcal{S}}_x$ then :

Interchange $q \in \bar{\mathcal{S}}_x$ with $p \in \mathcal{C}_z$

$\bar{\mathcal{S}}_z := \bar{\mathcal{S}}_z \cup q$
$\bar{\mathcal{S}}_x := \bar{\mathcal{S}}_x \backslash q$
$\mathcal{C} := \mathcal{C} \backslash p \cup q$
If internal reordering is needed, do UIQ2 and UIQ1
If internal reordering isn't needed, UIQ1
Goto [7].

[7] *Update* $\bar{\mathcal{S}}$

[7.1] Remove from $\bar{\mathcal{S}}$ all the superbasic variables set to one of its bounds by LSBE.
[7.2] If QNM used, apply UR4 as many times as superbasic variables deleted.

[8] If $\bar{\mathcal{S}}_z \neq \emptyset$ do :

[8.1] Select a suitable $p \in \mathcal{C}$ to interchange with q.
[8.2] Updates
$\mathcal{C} := \mathcal{C} \backslash p \cup q$.
$\bar{\mathcal{S}} := \bar{\mathcal{S}} \backslash q \cup p$.
g_z,π as usual, using *a)* and *b)* of UR3.
If QNM used, perform UR3.

[9] *Reinversion of* Q Performed each NRI updatings of Q^{-1}
[10] Goto [1]
[11] STOP : optimal solution found.

this is the basic structure of the algorithm developed. However, some remarks must be made.

7.1. Influence of a basic columns reordering on UR3

At steps [6.2.1] , [6.3.1] and [6.4.2] a two–column permutation can be applied. It has been explained above how this internal interchange affects the update of the working basis. It can be proved that a basic column permutation doesn't affect the standard update UR3, so, no special update of R must be performed in this cases.

7.2. Management of the slacks

The algorithm described dispenses an special treatement to the slacks. First, in step [2], when a nonbasic variable to leave its active bound is being looked for, slacks are scanned before than arcs. If a slack is found to be a good candidate, then it becomes superbasic, but only temporarily, because it is interchanged with a nonkey arc at the end of the present iteration (step [8]). This strategy tries to improve the stability of the solution of the systems of equations with coefficient matrix Q introducing in it as many identity columns as possible.

7.3. Trial step $p_z = \pm e_q$

There is an alternative way to manage the variable q which has been relaxed from one of its bounds in step [2] and added to $\bar{\mathcal{S}}$. It consists in fixing the values of the remaining superbasic variables and performing a null–space step of the form :

$$p_z = \begin{pmatrix} p_{z_1} \\ \vdots \\ p_{z_s} \\ \hline p_{z_q} \end{pmatrix} = \begin{pmatrix} 0 \\ \\ \hline \pm 1 \end{pmatrix} \tag{38}$$

with $p_{z_q} = +1$ if the new superbasic variable has been relaxed from its lower bound and $p_{z_q} = -1$ otherwise. It can be proved easily that this step is a descent direction. The new component of the reduced gradient is equal to the Lagrange multiplier associated with variable q :

$$g_{z_q} = -\sigma_q \tag{39}$$

testing the descent condition :

$$g_z' p_z = -\sigma_q p_{z_q} < 0 \tag{40}$$

the last inequality holds if we have selected q properly.

Once computed $p^{\bar{\mathcal{B}}}$ from p_z a linesearch LSCF is made. If the result of linesearch is the maximal step length, then superbasic variable $s+1$ is removed from $\bar{\mathcal{S}}$ and added to $\bar{\mathcal{B}}$, so, the number of superbasic variables has not increased. If such a situation happens, the iteration performed is nothing different than a simplex pivot between superbasic $s+1$ and the basic variable associated with the maximal step length.

If linesearch provides an $\alpha^* < \bar{\alpha}$, then if the temporary superbasic variable is an arc, it remains in $\bar{\mathcal{S}}$. If it is a slack, then an interchange with a nonkey arc is done as in step [8].

7.4. Description of the pivot operations in terms of the network structure

The simultaneous presence in the network of arcs $\mathcal{C}_x$ and arcs $\bar{\mathcal{S}}_x$ makes the interpretation and the management of the pivot operation a bit more difficult than in the pure nonlinear network flow problems. In the pure nonlinear network problem, it can

be assured that if the flow of a basic arc p change due to a step in the superbasic arcs then, it exist at least one superbasic arc that forms cycle with arc p. However, in the presence of side constraints, it is possible for an arc $p \in \mathcal{B}$ to change its value without forming cycle with any superbasic arc $q \in \bar{\mathcal{S}}_x$. From (6) and (28), the step $p^{\mathcal{B}}$ can be expressed as :

$$p^{\mathcal{B}} = -B^{-1}\left(\, A^{\bar{\mathcal{S}}} \;\middle|\; 0 \,\right) p_z - B^{-1}\left(\, A^{\mathcal{C}} \;\middle|\; 0 \,\right)\left(Q^{-1}DB^{-1}\left(\, A^{\bar{\mathcal{S}}} \;\middle|\; 0 \,\right) - Q^{-1}T^{\bar{\mathcal{S}}}\right) p_z \tag{41}$$

From (41) it is clear that :

$$p_p^{\mathcal{B}} \neq 0 \Rightarrow \beta^p A^{\bar{\mathcal{S}}} \neq (0) \quad \textit{or} \quad \beta^p A^{\mathcal{C}} \neq (0)$$

where β^p is the row of B^{-1} associated with arc p. The nonkey arcs can also change its value without been in a superbasic cycle, as shown in the following expression, build also from (6) and (28) :

$$p^{\mathcal{C}} = -Q^{-1}\left(\left(\, T^{\bar{\mathcal{S}}} \;\middle|\; 0 \,\right) - DB^{-1}\left(\, A^{\bar{\mathcal{S}}} \;\middle|\; 0 \,\right)\right) p_z \tag{42}$$

In step [6.2.2], a nonkey arc forming cycle with the outgoing key arc is first searched. If such an arc doesn't exist, then a direct pivot between key arc p an superbasic arc q must be made. Therefore, this operation can be only be performed if arcs p and q form cycle. This condition always holds, as it will be proof. The pivot vector associated to variable q is :

$$w = \bar{B}^{-1}\bar{S}_q = \bar{B}^{-1}\left(\begin{array}{c} A_q \\ \hline T_q \end{array}\right) \tag{43}$$

using formula (6), (43) can be expressed as :

$$w = \left(\begin{array}{c} w^1 \\ \hline w^2 \end{array}\right) = \left(\begin{array}{c} B^{-1}A_q + B^{-1}C\left(Q^{-1}DB^{-1}A_q - Q^{-1}T_q\right) \\ Q^{-1}\left(T_q - DB^{-1}A_q\right) \end{array}\right) \tag{44}$$

we are assuming that doesn't exist nonkey cycles, so $B^{-1}C = 0$ and (44) can be exepressed as :

$$w = \left(\begin{array}{c} w^1 \\ \hline w^2 \end{array}\right) = \left(\begin{array}{c} B^{-1}A_q \\ Q^{-1}\left(T_q - DB^{-1}A_q\right) \end{array}\right) \tag{45}$$

comparing the first m elements of (43) and (45) we obtain :

$$\bar{\beta}^j A_q = \beta^j A_q \quad , \quad j = 1, \ldots, m \tag{46}$$

if q has been selected properly in step [6.1], its pivot element must be different from zero :

$$\bar{\beta}^p A_q \neq 0 \Rightarrow \beta^p A_q \neq 0$$

so, arcs p and q forms cycle, and a direct pivot is allowed.

If a superbasic slack q is selected in [6.1] to become basic, and the variable p is an arc from the spanning tree , then, it will be necessary to perform a key non–key column interchange before the slack q has been entered in $\bar{\mathcal{B}}$. It can be assured that this previous interchange will be always possible because of the expression of the pivot element associated with slack q. If variable q has been selected to become basic, its pivot element must be different from zero. The pivot vector associated with slack q is :

$$w = \bar{B}^{-1}\bar{S}_q = \bar{B}^{-1}\begin{pmatrix} 0 \\ \hline e_q \end{pmatrix} \tag{47}$$

and the component associated with arc p is :

$$w_p = -\beta^p C Q^{-1} e_q \tag{48}$$

So,

$$w_p \neq 0 \Rightarrow \beta^p C \neq 0$$

, that is, there exists nonkey arcs forming cycle with arc p.

8. Examples

The numerical results for two problems with the same set of constraints and two differents objective functions. The dimensions of the problem are :

Number of arcs : $n = 228$

Number of nodes : $m = 49$

Number of s.c.: $t = 24$

8.1. Problem 1

Objective function : Cheap quadratic function.

	it. Ph.0	it. Ph.1	it. Ph.2	calls $f(x)$	$\mathcal{C}_x$ Ph.1	$\mathcal{C}_x$ Ph.2	$\mathcal{C}_z$ Ph.1	$\mathcal{C}_z$ Ph.2	$\bar{\mathcal{S}}$	$f(x)$ it.	t (sec.)
TNM	82	13	129	92	12	8	12	16	0	0.7	8.5
QNM	82	13	112	86	12	8	12	16	0	0.7	9.7

8.2. Problem 2

Objective function : Costly polynomial function

	it. Ph.0	it. Ph.1	it. Ph.2	calls $f(x)$	$\mathcal{C}_x$ Ph.1	$\mathcal{C}_x$ Ph.2	$\mathcal{C}_z$ Ph.1	$\mathcal{C}_z$ Ph.2	$\bar{\mathcal{S}}$	$f(x)$ it.	t (sec.)
TNM	82	13	205	559	12	18	12	6	15	2.7	27.1
QNM	82	13	191	338	12	18	12	6	9	1.7	17.5

BIBLIOGRAPHY

[1] D.P. Bertsekas,"Constrained optimization and Lagrange multiplier methods", (Academic Press, London, 1982).

[2] G.H. Bradley, G.G. Brown and G.W. Graves,"Design and implementation of large scale transshipment algorithms" Management Science 24 (1977) 1–34.

[3] R.S. Dembo,"A primal truncated newton algorithm with application to large-escale nonlinear network optimization", Mathematical Programming Studies 31 (1987) 43–71.

[4] R.S. Dembo and T. Steihaug, "Truncated–Newton algorithms for large–scale unconstrained optimization", Mathematical Programming 26 (1983) 190–212.

[5] E. Hellerman and D. Rarick,"Reinversion with the preassigned pivot procedure", Mathematical Programming 1 (1971) 195–216.

[6] J.L. Kennington and R.V. Helgason,"Algorithm for network programming",(John Wiley & Sons, New York, 1980).

[7] B.A. Murtagh and M.A. Saunders,"Large–scale linearly constrained optimization", Mathematical Programming 14 (1978) 41–72.

[8] Ph.L. Toint and D.Tuyttens,"On large scale nonlinear network optimization", Mathematical Programming 48 (1990) 125–159.