ON THE EFFICIENCY OF THE ε-SUBGRADIENT METHODS OVER NONLINEARLY CONSTRAINED NETWORKS

E. Mijangos,[1]
[1] *University of the Basque Country, Department of Applied Mathematics, Statistics and Operations Research, Spain, eugenio.mijangos@ehu.es*

Abstract The efficiency of the network flow techniques can be exploited in the solution of nonlinearly constrained network flow problems by means of approximate subgradient methods. In particular, we consider the case where the side constraints (non-network constraints) are convex. We propose to solve the dual problem by using ε-subgradient methods given that the dual function is estimated by minimizing *approximately* a Lagrangian function with only network constraints. Such Lagrangian function includes the side constraints. In order to evaluate the efficiency of these ε-subgradient methods some of them have been implemented and their performance computationally compared with that of other well-known codes. The results are encouraging.

keywords: Nonlinear Programming, Approximate Subgradient Methods, Network Flows.

1. Introduction

Consider the nonlinearly constrained network flow problem **(NCNFP)**

$$\begin{aligned} \underset{x}{\text{minimize}} \quad & f(x) && (1)\\ \text{subject to} \quad & x \in \mathcal{F} && (2)\\ & c(x) \leq 0, && (3) \end{aligned}$$

where:

- The set $\mathcal{F}$ is

$$\mathcal{F} = \{x \in \mathbf{R}^n \mid Ax = b,\ 0 \leq x \leq \overline{x}\},$$

where A is a node-arc incidence $m \times n$-matrix, b is the production/demand m-vector, x are the flows on the arcs of the network represented by A, and $\overline{x}$ are the capacity bounds imposed on the flows of each arc.

Please use the following format when citing this chapter:
Author(s) [insert Last name, First-name initial(s)], 2006, in IFIP International Federation for Information Processing, Volume 199, System Modeling and Optimization, eds. Ceragioli F., Dontchev A., Furuta H., Marti K., Pandolfi L., (Boston: Springer), pp. [insert page numbers].

- The side constraints (3) are defined by $c : \mathbf{R}^n \rightarrow \mathbf{R}^r$, such that $c = [c_1, \cdots, c_r]^t$, where $c_i(x)$ is linear or nonlinear and twice continuously differentiable on the feasible set $\mathcal{F}$ for all $i = 1, \cdots, r$.

- $f : \mathbf{R}^n \rightarrow \mathbf{R}$ is nonlinear and twice continuously differentiable on $\mathcal{F}$.

Many nonlinear network flow problems (in addition to the balance constraints on the nodes and the capacity constraints on the arc flows) have nonlinear side constraints. These are termed nonlinearly constrained network flow problems, **NCNFP**. In recent works [11, 12], **NCNFP** has been solved using partial augmented Lagrangian methods with quadratic penalty function and superlinear-order multiplier estimates (ALM).

In this work we focus on the primal problem **NCNFP** and its dual problem

$$\text{maximize} \qquad q(\mu) = \min_{x \in \mathcal{F}} l(x, \mu) = \min_{x \in \mathcal{F}} \{f(x) + \mu^t c(x)\} \tag{4}$$

$$\text{subject to:} \qquad \mu \in \mathcal{M}, \tag{5}$$

where $\mathcal{M} = \{\mu \mid \mu \geq 0,\ q(\mu) > -\infty\}$. We assume throughout this paper that the constraint set $\mathcal{M}$ is closed and convex, and q is continuous on $\mathcal{M}$, and for every $\mu \in \mathcal{M}$ some vector $x(\mu)$ that minimizes $l(x, \mu)$ over $x \in \mathcal{F}$ can be calculated, yielding a subgradient $c(x(\mu))$ of q at μ. We propose to solve **NCNFP** by using primal-dual methods, see [2].

The minimization of the Lagrangian function $l(x, \mu)$ over $\mathcal{F}$ can be performed by means of efficient techniques specialized for networks, see [21].

Since $q(\mu)$ is approximately computed, we consider *approximate subgradient methods* [13] in the solution of this problem. Author's purpose is to improve the efficiency obtained by using the multiplier methods with *asymptotically exact minimization* [11, 12]. Moreover, these methods allow us to solve problems of the kind of **NCNFP** where the dual function might be nondifferentiable in spite of having a differentiable Lagrangian function, as this could happen if the conditions given by the Proposition 6.1.1 in [2] are not fulfilled. The basic difference between these methods and the classical subgradient methods is that they replace the subgradients with inexact subgradients.

Different ways of computing the stepsize in the approximate subgradient methods have been considered. The diminishing stepsize rule (DSR) suggested by Correa and Lemaréchal in [4]. A dynamically chosen stepsize rule based on an estimation of the optimal value of the dual function by means of an adjustment procedure (DSAP) similar to that suggested by Nedić and Bertsekas in [18] for incremental subgradient methods. A dynamically chosen stepsize whose estimate of the optimal value of the dual function is based on the relaxation level-control algorithm (DSRLC) designed by Brännlund in [3] and analyzed by Goffin and Kiwiel in [9]. The convergence of these methods was studied in the cited papers for the case of exact subgradients. The convergence

of the corresponding approximate (inexact) subgradient methods is analyzed in [13], see also [10].

The main aim of this work is to evaluate the efficiency of the approximate subgradient methods when we use DSR, DSAP, and DSRLC over **NCNFP** problems, for which we compare it with that of the augmented Lagrangian method (ALM) when it uses superlinear-order estimates [11, 12], and with that of the well-known codes filterSQP [7] and MINOS [17]. Also, we compare the quality of the computed solution by the ε-subgradient methods.

This paper is organized as follows: Section 2 presents the approximate subgradient methods; Section 3, the solution to the nonlinearly constrained network flow problem; and Section 4 puts forward the numerical tests.

2. Approximate subgradient methods

When, as happens in this work, for a given $\mu \in \mathcal{M}$, the dual function value $q(\mu)$ is calculated by minimizing approximately $l(x, \mu)$ over $x \in \mathcal{F}$ [see (4)], the subgradient obtained, as well as the value of $q(\mu)$, will involve an error.

In order to put forward such methods, it is useful to introduce a notion of approximate subgradient [2, 20]. In particular, given a scalar $\varepsilon \geq 0$ and a vector $\overline{\mu}$ with $q(\overline{\mu}) > -\infty$, we say that c is an *ε-subgradient at $\overline{\mu}$* if

$$q(\mu) \leq q(\overline{\mu}) + \varepsilon + c^t(\mu - \overline{\mu}), \qquad \forall \mu \in \mathbf{R}^r. \tag{6}$$

The set of all ε-subgradients at $\overline{\mu}$ is called the *ε-subdifferential at $\overline{\mu}$* and is denoted by $\partial_\varepsilon q(\overline{\mu})$. Note that every subgradient at a given point is also an ε-subgradient for all $\varepsilon > 0$. Generally, however, an ε-subgradient need not be a subgradient, unless $\varepsilon = 0$.

An approximate subgradient method is defined by

$$\mu^{k+1} = [\mu^k + s_k c^k]^+, \tag{7}$$

where c^k is an ε_k-subgradient at μ^k, $[\cdot]^+$ denotes the projection on the closed convex set $\mathcal{M}$, and s_k is a positive stepsize.

In our context, we minimize approximately $l(x, \mu^k)$ over $x \in \mathcal{F}$, thereby obtaining a vector $x^k \in \mathcal{F}$ with

$$l(x^k, \mu^k) \leq \inf_{x \in \mathcal{F}} l(x, \mu^k) + \varepsilon_k. \tag{8}$$

As is shown in [2, 13], the corresponding constraint vector, $c(x^k)$, is an ε_k-subgradient at μ^k. If we denote $q_{\varepsilon_k}(\mu^k) = l(x^k, \mu^k)$, by definition of $q(\mu^k)$ and using (8) we have

$$q(\mu^k) \leq q_{\varepsilon_k}(\mu^k) \leq q(\mu^k) + \varepsilon_k \qquad \forall k. \tag{9}$$

2.1 Stepsize rules

Throughout this section, we use the notation

$$q^* = \sup_{\mu \in \mathcal{M}} q(\mu), \quad \mathcal{M}^* = \{\mu \in \mathcal{M} \mid q(\mu) = q^*\},$$

and $\|\cdot\|$ denotes the standard Euclidean norm.

In this work, three kinds of stepsize rules have been considered.

2.1.1 Diminishing stepsize rule (DSR). The convergence of the subgradient method using a diminishing stepsize was shown by Correa and Lemaréchal, see [4]. Next, we consider the special case where c^k is an ε_k-subgradient.

In a recent work [13], the following proposition is proved.

PROPOSITION 1 *Let the optimal set $\mathcal{M}^*$ be nonempty. Also, assume that the sequences $\{s_k\}$ and $\{\varepsilon_k\}$ are such that*

$$s_k > 0, \ \sum_{k=0}^{\infty} s_k = \infty, \ \sum_{k=0}^{\infty} s_k^2 < \infty, \ \sum_{k=0}^{\infty} s_k \varepsilon_k < \infty. \tag{10}$$

Then, the sequence $\{\mu^k\}$, generated by the ε-subgradient method, where $c^k \in \partial_{\varepsilon_k} q(\mu^k)$ (with $\{\|c^k\|\}$ bounded), converges to some optimal solution.

An example of such a stepsize is

$$s^k = 1/\widehat{k}, \tag{11}$$

for $\widehat{k} = \lfloor k/m \rfloor + 1$. In this work we use by default $m = 5$.

An interesting alternative for the ordinary subgradient method is the *dynamic stepsize rule*

$$s_k = \gamma_k \frac{q^* - q(\mu^k)}{\|c^k\|^2}, \tag{12}$$

with $c^k \in \partial q(\mu^k)$ and $0 < \underline{\gamma} \leq \gamma_k \leq \overline{\gamma} < 2$, which was introduced by Poljak in [19] (see also Shor [20]).

Unfortunately, in most practical problems q^* and $q(\mu^k)$ are unknown. Then, the latter can be approximated by $q_{\varepsilon_k}(\mu^k) = l(x^k, \mu^k)$ and q^* replaced with an estimate q_{lev}^k. This leads to the stepsize rule

$$s_k = \gamma_k \frac{q_{lev}^k - q_{\varepsilon_k}(\mu^k)}{\|c^k\|^2}, \tag{13}$$

where $c^k \in \partial_{\varepsilon_k} q(\mu^k)$ is bounded for $k = 0, 1, \ldots$.

2.1.2 Dynamic stepsize with adjustment procedure (DSAP). An option to estimate q^* is to use the *adjustment procedure* suggested by Nedić and Bertsekas [18], but fitted for the ε-subgradient method, its convergence is analyzed by Mijangos in [13], see also [10].

In this procedure q_{lev}^k is the best function value achieved up to the kth iteration, in our case $\max_{0\le j\le k} q_{\varepsilon_j}(\mu^j)$, plus a positive amount δ_k, which is adjusted according to algorithm's progress.

The adjustment procedure obtains q_{lev}^k as follows:

$$q_{lev}^k = \max_{0\le j\le k} q_{\varepsilon_j}(\mu^j) + \delta_k,$$

and δ_k is updated according to

$$\delta_{k+1} = \begin{cases} \rho\delta_k, & \text{if } q_{\varepsilon_{k+1}}(\mu^{k+1}) \ge q_{lev}^k, \\ \max\{\beta\delta_k, \delta\}, & \text{if } q_{\varepsilon_{k+1}}(\mu^{k+1}) < q_{lev}^k, \end{cases}$$

where δ_0, δ, β, and ρ are fixed positive constants with $\beta < 1$ and $\rho \ge 1$.

2.1.3 Dynamic stepsize with relaxation-level control (DSRLC). Another choice to compute an estimate q_{lev}^k for (13) is to use a dynamic stepsize rule with relaxation-level control, which is based on the algorithm given by Brännlund [3], whose convergence was proved by Goffin and Kiwiel in [9] for $\varepsilon_k = 0$ for all k.

Mijangos in [13] has fitted this method to the dual problem of **NCNFP** (4-5) for $\{\varepsilon_k\} \to 0$ and analized its convergence, see also [10].

In this case, in contrast to the *adjustment procedure*, q^* is estimated by q_{lev}^k, which is a target level that is updated only if a sufficient ascent is detected or when the path long done from the last update exceeds a given upper bound B.

ALGORITHM 1

Step 0 (*Initialization*): Select μ^0, $\delta_0 > 0$, and $B > 0$.

Set $\sigma_0 = 0$ and $q_{rec}^{-1} = \infty$.

Set $k = 0$, $l = 0$ and $k(l) = 0$, where $k(l)$ will denote the iteration k when the lth update of q_{lev}^k occurs. Then $k(l) = k$ will be set.

Step 1 (*Function evaluation*): Compute $q_{\varepsilon_k}(\mu^k)$ and $c^k \in \partial_{\varepsilon_k} q(\mu^k)$.

If $q_{\varepsilon_k}(\mu^k) > q_{rec}^{k-1}$, set $q_{rec}^k = q_{\varepsilon_k}(\mu^k)$.

Otherwise set $q_{rec}^k = q_{rec}^{k-1}$.

Step 2 (*Stopping rule*): If $\|c^k\| = 0$, terminate with $\mu^* = \mu^k$.

Step 3 (*Sufficient ascent detection*): If $q_{\varepsilon_k}(\mu^k) \ge q_{rec}^{k(l)} + \frac{1}{2}\delta_k$, set $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \delta_l$, $l := l + 1$, and go to Step 5.

Step 4 (*Oscillation detection*): If $\sigma_k > B$, set $k(l+1) = k$, $\sigma_k = 0$, $\delta_{l+1} = \frac{1}{2}\delta_l$, $l := l+1$.

Step 5 (*Iterate update*): Set $q_{lev}^k = q_{rec}^{k(l)} + \delta_l$. Choose $\gamma \in [\underline{\gamma}, \overline{\gamma}]$ and compute μ^{k+1} by means of (7) with the stepsize s_k obtained by (13).

Step 6 (*Path long update*): Set $\sigma_{k+1} = \sigma_k + s_k\|c^k\|$, $k := k+1$, and go to Step 1.

Note that q_{rec}^k keeps the record of the highest value attained by the iterates that are generated so far; i.e., $q_{rec}^k = \max_{0\leq j\leq k} q_{\varepsilon_j}(\mu^j)$. Moreover, the algorithm uses the same target level $q_{lev}^k = q_{rec}^{k(l)} + \delta_l$ for $k = k(l), k(l)+1, k(l+2), \ldots, k(l+1)-1$. In [13] we analize the convergence of the ε-subgradient method with the stepsize (13) for q_{lev} given by this algorithm.

3. Solution to NCNFP

An algorithm is given below for solving **NCNFP**. This algorithm uses the approximate subgradient method described in Section 2.

The value of the dual function $q(\mu^k)$ is estimated by minimizing approximately $l(x, \mu^k)$ over $x \in \mathcal{F}$ (the set defined by the network constraints) so that the optimality tolerance, τ_x^k, becomes more rigorous as k increases; i.e., the minimization will be *asymptotically exact* [1]. In other words, we set $q_{\varepsilon_k}(\mu^k) = l(x^k, \mu^k)$, where x^k minimizes approximately the nonlinear network subproblem $\mathbf{NNS_k}$

$$\underset{x\in\mathcal{F}}{\text{minimize}}\ l(x, \mu^k)$$

in the sense that this minimization stops when we obtain a x^k such that

$$\|Z^t \nabla_x l(x^k, \mu^k)\| \leq \tau_x^k$$

where $\lim_{k\to\infty} \tau_x^k = 0$ and Z represents the reduction matrix whose columns form a base of the null subspace of the subspace generated by the rows of the matrix of active network constraints of this subproblem, see [16]. Let $\overline{x}^k$ be the minimizer of this subproblem approximated by x^k. Then, it can be proved (see [13]) that there exists a positive w, such that $l(x^k, \mu^k) \leq l(\overline{x}^k, \mu^k) + w\tau_x^k$ for $k = 1, 2, \ldots$. If we set $\varepsilon_k = \omega\tau_x^k$, this inequality becomes (8). Moreover, as

$$\tau_x^{k+1} = \alpha\tau_x^k, \qquad \text{for a fixed} \quad \alpha \in (0,1),$$

then $\sum_{k=1}^{\infty} \varepsilon_k < \infty$, and so $\lim_{k\to\infty} \varepsilon_k = 0$. Consequently, we can denote $q_{\varepsilon_k} = l(x^k, \mu^k)$, which holds the inequality (9), and we may use the methods described in Section 2. In this work, $\alpha = 10^{-1}$ by default. Note that in this case, $\varepsilon_k = \tau_x^k\omega = 10^{-k-1}\omega$.

ALGORITHM 2 (APPROXIMATE SUBGRADIENT METHOD FOR **NCNFP**)

Step 0 *Initialize.* Set $k = 1$, N_{max}, τ_x^1, ϵ_μ and τ_μ. Set $\mu^1 = 0$.

Step 1 *Compute* the dual function estimate, $q_{\varepsilon_k}(\mu^k)$, by solving $\mathbf{NNS_k}$, so that if $\|Z^t \nabla_x l(x^k, \mu^k)\| \leq \tau_x^k$, then $x^k \in \mathcal{F}$ is an approximate solution, $q_{\varepsilon_k}(\mu^k) = l(x^k, \mu^k)$, and $c^k = c(x^k)$ is an ε_k-subgradient of q in μ^k.

Step 2 *Check the stopping rules* for μ^k.

T_1: Stop if $\max\limits_{i=1,\ldots,r} \left\{ \dfrac{(c_i^k)^+}{1 + (c_i^k)^+} \right\} < \tau_\mu$, where $(c_i^k)^+ = \max\{0, c_i(x^k)\}$.

T_2: Stop if $\left| \dfrac{q^k - (q^{k-1} + q^{k-2} + q^{k-3})/3}{1 + q^k} \right| < \epsilon_\mu$, where $q^n = q_{\varepsilon_n}(\mu^n)$.

T_3: Stop if $\dfrac{1}{4} \sum\limits_{i=0}^{3} \|\mu^{k-i} - \mu^{k-i-1}\|_\infty < \epsilon_\mu$.

T_4: Stop if k reaches a prefixed value N_{max}.

If μ^k fulfils one of these tests, then it is optimal, and the algorithm stops. Without a duality gap, (x^k, μ^k) is a primal-dual solution.

Step 3 *Update* the estimate μ^k by means of the iteration

$$\mu_i^{k+1} = \begin{cases} \mu_i^k + s_k c_i^k, & \text{if } \mu_i^k + s_k c_i^k > 0 \\ 0, & \text{otherwise} \end{cases}$$

where s_k is computed using some stepsize rule. Go to Step 1.

In Step 0, for the checking of the stopping rules, $\tau_\mu = 10^{-5}$, $\epsilon_\mu = 10^{-5}$ and $N_{max} = 200$ have been taken. In addition, $\tau_x^0 = 10^{-1}$ by default.

Step 1 of this algorithm is carried out by the code PFNL, described in [14] (downloadable from website http://www.ehu.es/~mepmifee/).

In Step 2, alternative heuristic tests have been used for practical purposes. T_1 checks the feasibility of x^k, as if it is feasible the duality gap is zero, and then (x^k, μ^k) is a primal-dual solution for **NCNFP**. T_2 and T_3 mean that μ does not improve for the last N iterations. Note that $N = 4$.

To obtain s_k in Step 3, we have used the iteration (7) with the three rules considered in Section 2 for computing the stepsize:

▷ Diminishing stepsize rule DSR given by (11), which holds (10), for the ϵ_k given above.

▷ Dynamic stepsize with adjustment procedure DSAP, using $\rho = 2$, $\beta = 1/\rho$, $\gamma_k = 1$ for all k, $\delta_0 = 0.05\|(c^1)^+\|$, and $\delta = 10^{-5}|l(x_0, \mu_0)|$, where x^0 is the initial feasible point for Step 1 and $k = 0$.

▷ Dynamic stepsize with relaxation level-control DSRLC, replacing B in Algorithm 1 with $B_l = \max\{\overline{B}, B/l\}$ when an oscillation is detected, for $B = 10^{-3}\|x^1 - x^0\|$ and $\overline{B} = 0.01$. As can be seen, $\sum_{l=1}^{\infty} B_l = \infty$. In addition, we set $\gamma_k = 1$ for all k, $\delta = 10^{-5}|l(x_0, \mu_0)|$, $\delta_0 = 0.5\|(c^1)^+\|B$

The values given above have been heuristically chosen. The implementation in Fortran-77 of the previous algorithm, termed PFNRN05, was designed to solve large-scale nonlinear network flow problems with nonlinear side constraints.

4. Numerical tests

In order to obtain an evaluation of PFNRN05, some computational tests are performed, which consist in solving nonlinear network flow problems with nonlinear side constraints using this code and comparing the results with those obtained using PFNRN (with ALM), filterSQP and MINOS (see Section 1). These last two solvers are available on the NEOS server [5] with AMPL input [8]. (See the site `http://www-neos.mcs.anl.gov/`.) PFNRN is executed on a Sun Sparc 10/41 work station under UNIX (which has a similar speed to that of the NEOS machines).

Table 1. Test problems.

problem	# arcs	# nodes	# side const.	# actives	# sb. arcs
D12e2	1524	360	180	5	75
D13e2	1524	360	360	10	89
D14e2	1524	360	36	31	151
D12n1	1524	360	180	22	685
D13n1	1524	360	360	38	681
D14n1	1524	360	36	31	596
D21e2	5420	1200	120	3	30
D22e2	5420	1200	120	18	45
D23e2	5420	1200	120	17	238
D31e1	4008	501	5	1	63
D31e2	4008	501	5	1	60

The problems used in these tests were created by means of the following DIMACS-random-network generators: Rmfgen and Gridgen, see [6]. These generators provide linear flow problems in networks without side constraints. The inequality nonlinear side constraints for the DIMACS networks were generated through the *Dirnl* random generator described in [14]. The last two

Table 2. Comparison of the computed solution.

	DSR		DSAP		DSRLC	
Problem	e/f^*	$\|c\|_\infty$	e/f^*	$\|c\|_\infty$	e/f^*	$\|c\|_\infty$
D12e2	10^{-4}	10^{-3}	0.	10^{-11}	0.	10^{-11}
D13e2	10^{-4}	10^{-3}	10^{-7}	10^{-8}	10^{-7}	10^{-8}
D14e2	10^{-4}	10^{-3}	10^{-6}	10^{-6}	10^{-6}	10^{-6}
D12n1	10^{-6}	10^{-3}	0.	10^{-8}	10^{-7}	10^{-6}
D13n1	10^{-6}	10^{-3}	10^{-7}	10^{-6}	10^{-7}	10^{-7}
D14n1	10^{-4}	10^{-5}	10^{-5}	10^{-6}	0.	10^{-10}
D21e2	10^{-4}	10^{-2}	10^{-7}	10^{-7}	0.	10^{-11}
D22e2	10^{-3}	10^{-2}	10^{-6}	10^{-6}	0.	10^{-7}
D23e2	10^{-5}	10^{-1}	10^{-6}	10^{-7}	10^{-6}	10^{-7}
D31e1	10^{-7}	10^{-7}	0.	10^{-11}	0.	10^{-15}
D31e2	10^{-8}	10^{-6}	0.	10^{-14}	0.	10^{-7}

Table 3. Comparison of the efficiency.

Problem	filterSQP	MINOS	ALM	DSR	DSAP	DSRLC
D12e2	3.3	0.5	1.0	0.3	0.6	0.4
D13e2	5.0	0.7	2.3	0.6	0.5	0.6
D14e2	22.7	–	18.7	4.1	3.3	4.2
D12n1	409.6	31.1	46.2	48.3	58.3	39.6
D13n1	560.8	47.9	60.5	73.3	54.0	46.6
D14n1	–	–	162.3	330.3	178.3	103.3
D21e2	38.7	3.7	2.2	1.4	1.4	1.6
D22e2	63.5	6.8	5.4	2.5	1.9	1.9
D23e2	112.4	239.3	20.0	5.8	6.2	5.9
D31e1	28.7	45.6	2.0	1.6	1.6	1.6
D31e2	22.7	14.1	1.3	0.9	1.0	1.0

letters indicate the type of objective function that we have used: Namur functions, **n***, and EIO1 functions, **e***. The EIO1 family creates problems with a moderate number of superbasic variables (i.e., dimension of the null space) at the solution (# sb. arcs). By contrast, the Namur functions [21] generates a high number of superbasic arcs at the optimizer, see Table 1. More details about these problems can be found in [14, 15].

In Table 2, e/f^* represents the relative error in the computation of the optimum value of the objective function, whereas $\|\overline{c}\|_\infty$ represents the maximum violation of the side constraints in the optimal solution; that is, it offers information about the feasibility of this solution and, hence, about its duality gap. The results point out that the quality of the solution computed by PFNRN05 when it uses DSR is lower than that obtained when using dynamic stepsizes, such as DSAP or DSRLC.

In Table 3, for each method used to compute the stepsize the efficiency is evaluated by means of the run-times in CPU-seconds. The efficiency for the three stepsizes is very similar and, for these tests, the subgradient methods were more efficient than the quadratic multiplier method, ALM (see Section 1), filterSQP, and MINOS. Moreover, with default values, filterSQP was more accurate than MINOS.

These results encourage to carry out further experimentation, which also includes real problems, and to analyze more carefully the influence of some parameters over the performance of this code.

References

[1] D.P. Bertsekas. *Constrained Optimization and Lagrange Multiplier Methods*. Academic Press, New York, 1982.

[2] D.P. Bertsekas, *Nonlinear Programming*. 2nd ed. Athena Scientific, Belmont, Massachusetts, 1999.

[3] U. Brännlund. On relaxation methods for nonsmooth convex optimization. Doctoral Thesis, Royal Institute of Technology, Stockholm, Sweden, 1993.

[4] R. Correa, C. Lemarechal. Convergence of some algorithms for convex minimization. *Mathematical Programming*, 62:261–275, 1993.

[5] J. Czyzyk, M.P. Mesnier, J. Moré. The NEOS server. *IEEE Computational Science and Engineering*, 5(3):68–75, 1998.

[6] DIMACS. The first DIMACS international algorithm implementation challenge : The bench-mark experiments. Technical Report, DIMACS, New Brunswick, NJ, USA, 1991.

[7] R. Fletcher and S. Leyffer. User manual for filterSQP, University of Dundee Numerical Analysis Report NA\181, 1998.

[8] R. Fourer, D.M. Gay, B.W. Kernighan. *AMPL a modelling language for mathematical programming*. Boyd and Fraser Publishing Company, Danvers, MA 01293, USA, 1993.

[9] J.L. Goffin, K. Kiwiel. Convergence of a simple subgradient level method. *Mathematical Programming*, 85:207–211, 1999.

[10] K. Kiwiel. Convergence of approximate and incremental subgradient methods for convex optimization. *SIAM Journal on Optimization*, 14(3):807–840, 2004.

[11] E. Mijangos. An implementation of Newton-like methods on nonlinearly constrained networks. *Computers and Operations Research*, 32(2):181–199, 2004.

[12] E. Mijangos. An efficient method for nonlinearly constrained networks. *European Journal of Operational Research*, 161(3):618–635, 2005.

[13] E. Mijangos, Approximate subgradient methods for nonlinearly constrained network flow problems. To appear in *Journal of Optimization Theory and Applications*, 128(1), 2006.

[14] E. Mijangos, N. Nabona. The application of the multipliers method in nonlinear network flows with side constraints. Technical Report 96/10, Dept. of Statistics and Operations Research, Universitat Politècnica de Catalunya, Barcelona, Spain, 1996.

[15] E. Mijangos, N. Nabona. On the first-order estimation of multipliers from Kuhn-Tucker systems. *Computers and Operations Research*, 28:243–270, 2001.

[16] B.A. Murtagh, M.A. Saunders. Large-scale linearly constrained optimization. *Mathematical Programming*, 14:41–72, 1978.

[17] B.A. Murtagh, M.A. Saunders. MINOS 5.5. User's guide. Report SOL 83-20R, Department of Operations Research, Stanford University, Stanford, CA, USA, 1998.

[18] A. Nedić, D.P. Bertsekas. Incremental subgradient methods for nondifferentiable optimization. *SIAM Journal on Optimization*, 12:109–138, 2001.

[19] B.T. Poljak. Minimization of unsmooth functionals, *Z. Vyschisl. Mat. i Mat. Fiz.*, 9:14–29, 1969.

[20] N.Z. Shor. *Minimization methods for nondifferentiable functions*. Springer-Verlag, Berlin, 1985.

[21] Ph.L. Toint, D. Tuyttens. On large scale nonlinear network optimization. *Mathematical Programming*, 48:125–159, 1990.