Large-scale active-set box-constrained optimization method with spectral projected gradients

Ernesto G. Birgin [*] José Mario Martínez [†]

December 11, 2001

Abstract

A new active-set method for smooth box-constrained minimization is introduced. The algorithm combines an unconstrained method, including a new line-search which aims to add many constraints to the working set at a single iteration, with a recently introduced technique (spectral projected gradient) for dropping constraints from the working set. Global convergence is proved. A computer implementation is fully described and a numerical comparison assesses the reliability of the new algorithm.

Keywords: Box-constrained minimization, numerical methods, active-set strategies, Spectral Projected Gradient.

1 Introduction

The problem considered in this paper consists in the minimization of a smooth function $f : I\!R^n \to I\!R$ with bounds on the variables. The feasi-

[*]Department of Computer Science IME-USP, University of São Paulo, Rua do Matão 1010, Cidade Universitária, 05508-090, São Paulo SP, Brazil. This author was supported by PRONEX-Optimization 76.79.1008-00, FAPESP (Grants 99/08029-9 and 01/04597-4) and CNPq (Grant 300151/00-4). e-mail: egbirgin@ime.usp.br

[†]Department of Applied Mathematics IMECC-UNICAMP, University of Campinas, CP 6065, 13081-970 Campinas SP, Brazil. This author was supported by PRONEX-Optimization 76.79.1008-00, FAPESP (Grant 01/04597-4), CNPq and FAEP-UNICAMP. e-mail: martinez@ime.unicamp.br

ble set Ω is defined by

$$\Omega = \{x \in I\!R^n \mid \ell \leq x \leq u\}. \tag{1}$$

Box-constrained minimization algorithms are used as subalgorithms for solving the subproblems that appear in many augmented Lagrangian and penalty methods for general constrained optimization. See [11, 12, 16, 17, 18, 19, 20, 21, 26, 28, 31]. A very promising novel application is the reformulation of equilibrium problems. See [1] and references therein. The methods introduced in [11] and [26] are of trust-region type. For each iterate $x_k \in \Omega$, a quadratic approximation of f is minimized in a trust-region box. If the objective function value at the trial point is sufficiently smaller than $f(x_k)$, the trial point is accepted. Otherwise, the trust region is reduced. The difference between [11] and [26] is that, in [11], the trial point is in the face defined by a "Cauchy point", whereas in [26] the trial point is obtained by means of a specific box-constrained quadratic solver, called QUACAN. See [2, 15, 23, 25, 31] and [13] (p. 459). Other trust-region methods for box-constrained optimization have been introduced in [3, 29].

QUACAN is an active-set method that uses conjugate gradients within the faces, approximate internal-face minimizations, projections to add constraints to the active set and an "orthogonal-to-the-face" direction to leave the current face when an approximate minimizer in the face is met. In [17] a clever physical interpretation for this direction was given.

Numerical experiments in [16] suggested that the efficiency of the algorithm [26] relies, not in the trust-region strategy, but in the strategy of QUACAN for dealing with constraints. This motivated us to adapt the strategy of QUACAN to general box-constrained problems. Such adaptation involves two main decisions. On one hand, one needs to choose an unconstrained minimization algorithm to deal with the objective function within the faces. On the other hand, it is necessary to define robust and efficient strategies to leave faces and to add active constraints. Attempts for the first decision have been made in [6] and [10]. In [10] a secant multipoint minimization algorithm is used and in [6] the authors use the second-order minimization algorithm of Zhang and Xu [36].

In this paper we adopt the leaving-face criterion of [6], that employs the spectral projected gradients defined in [7, 8]. See, also, [4, 5, 32, 33, 34].

For the internal minimization in the faces we introduce a new general algorithm with a line search that combines backtracking and extrapolation. The compromise in every line-search algorithm is between accuracy in the localization of the one-dimensional minimizer and economy in terms of functional evaluations. Backtracking-like line-search algorithms are cheap but, sometimes, tend to generate excessively small steps. For this reason, backtracking is complemented with a simple extrapolation procedure here. The direction chosen at each step is arbitrary, provided that an angle condition is satisfied.

In the implementation described in this paper, we suggest to choose the direction using the truncated-Newton approach. This means that the search vector is an approximate minimizer of the quadratic approximation of the function in the current face. We use conjugate gradients to find this direction, so the first iterate is obviously a descent direction, and this property is easily monitorized through successive conjugate gradient steps.

The present research is organized as follows. In Section 2 we describe an "unconstrained" minimization algorithm that deals with the minimization of a function on a box. The algorithm uses the new line-search technique. Due to this technique it is possible to prove that either the method finishes at a point on the boundary where, perhaps, many constraints are added, or it converges to a point in the box where the gradient vanishes. The box-constrained algorithm is described in Section 3. Essentially, we use the algorithm of Section 2 to work within the "current face" and spectral projected gradients [7] to leave constraints. The spectral projected gradient technique also allows one to leave many bounds and to add many others to the working set at a single iteration. This feature can be very important for large-scale calculations. In this section we prove the global convergence of the box-constrained algorithm. The computational description of the code (GENCAN) is given in Section 4. In Section 5 we show numerical experiments using the CUTE collection. In Section 6 we report experiments using some very large problems (up to 10^7 variables). Finally, in Section 6 we make final comments and suggest some lines for future research.

2 A general "unconstrained" algorithm

In this section we assume that $f : I\!R^{\bar{n}} \to I\!R$, $f \in C^1(I\!R^n)$ and

$$\mathcal{B} = \{x \in I\!R^{\bar{n}} \mid \bar{\ell} \leq x \leq \bar{u}\}.$$

The set $\mathcal{B}$ will represent each of the closed faces of Ω in Section 3. The dimension $\bar{n}$ in this section is the dimension of the reduced subspace of the Section 3 and the gradient of this section is composed by the derivatives with respect to free variables in Section 3. We hope that using the notation ∇f in this section will not lead to confusion.

From now on, we denote $g(x) = \nabla f(x)$ and $g_k = g(x_k) = \nabla f(x_k)$. Our objective is to define a general iterative algorithm that starts in the interior of $\mathcal{B}$ and, either converges to an unconstrained stationary point, or finishes in the boundary of $\mathcal{B}$ having decreased the functional value. This will be the algorithm used "within the faces" in the box-constrained method. Algorithm 2.1 is based on line searches with Armijo-like conditions and extrapolation. Given the current point x_k and a descent direction d_k, we finish the line search if $x_k + d_k$ satisfies a sufficient descent criterion and if the directional derivative is sufficiently larger than $\langle g(x_k), d_k \rangle$. If the sufficient descent criterion does not hold, we do backtracking. If we obtained sufficient descent but the increase of the directional derivative is not enough, we try extrapolation.

Let us explain why we think that this philosophy is adequate for large-scale box-constrained optimization.

1. Pure backtracking is enough for proving global convergence of many optimization algorithms. However, to accept the first trial point when it satisfies an Armijo condition can lead to very small steps in critical situations. Therefore, steps larger than the unity must be tried when some indicator says that this is worthwhile.

2. If the directional derivative $\langle g(x_k + d_k), d_k \rangle$ is sufficiently larger than $\langle g(x_k), d_k \rangle$ we consider that there is not much to decrease increasing the steplength in the direction of d_k and, so, we accept the unit steplength provided it satisfies the Armijo condition. This is reasonable since, usually, the search direction contains some amount of

second-order information that makes the unitary steplength desirable from the point of view of preserving a satisfactory order of convergence.

3. If the unitary steplength does not satisfy the Armijo condition, we do backtracking. In this case we judge that it is not worthwhile to compute gradients of the new trial points, which would be discarded if the point is not accepted.

4. Extrapolation is especially useful in large-scale problems, where it is important to try to add as many constraints as possible to the working set. So, we extrapolate in a rather greedy way, multiplying the steplength by a fixed factor while the function value decreases.

5. We think that the algorithm presented here is the most simple way in which extrapolation devices can be introduced with a reasonable balance between cost and efficiency. It is important to stress that this line search can be coupled with virtually any minimization procedure that computes descent directions.

For all $z \in I\!R^n$, the Euclidean projection of z onto a convex set S will be denoted $P_S(z)$. In this section, we denote $P(y) = P_{\mathcal{B}}(y)$. The symbol $\|\cdot\|$ represents the Euclidean norm throughout the paper.

Algorithm 2.1: Line-search based algorithm
The algorithm starts with $x_0 \in Int(\mathcal{B})$. The non-dimensional parameters $\gamma \in (0, 1/2)$, $\beta \in (0, 1)$, $\theta \in (0, 1)$, $N > 1$ and $0 < \sigma_1 < \sigma_2 < 1$ are given. We also use the small tolerances $\epsilon_{abs}, \epsilon_{rel} > 0$. Initially, we set $k \leftarrow 0$.

Step 1. *Computing the search direction*

Step 1.1 If $\|g_k\| = 0$, stop.

Step 1.2 Compute $d_k \in I\!R^n$ such that

$$\langle g_k, d_k \rangle \le -\theta \|g_k\| \|d_k\|. \tag{2}$$

Step 2. *Line-search decisions*

Step 2.1 Compute $\alpha_{\max} \leftarrow \max\{\alpha \ge 0 \mid [x_k, x_k + \alpha d_k] \subset \mathcal{B}\}$ and set $\alpha \leftarrow \min\{\alpha_{\max}, 1\}$.
If ($\alpha_{\max} > 1$, so $x_k + d_k \in Int(\mathcal{B})$) then go to Step 2.2

else go to Step 2.3.

Step 2.2 (At this point we have $x_k + d_k \in Int(\mathcal{B})$.)

$$\text{If } (f(x_k + d_k) \leq f(x_k) + \gamma\langle g_k, d_k\rangle) \text{ then} \tag{3}$$

$$\text{if } (\langle d_k, g(x_k + d_k)\rangle \geq \beta\langle g_k, d_k\rangle) \text{ then} \tag{4}$$

take $\alpha_k = 1$, $x_{k+1} = x_k + d_k$ and go to Step 5

else go to Step 3 (Extrapolation)

else go to Step 4 (Backtracking).

Step 2.3 (At this point we have $x_k + d_k \notin Int(\mathcal{B})$.)

If $(f(x_k + \alpha_{\max} d_k) < f(x_k))$ then

take $\alpha_k \geq \alpha_{\max}$ and $x_{k+1} = P(x_k + \alpha_k d_k) \in \mathcal{B} - Int(\mathcal{B})$ such that $f(x_{k+1}) \leq f(x_k + \alpha_{\max} d_k)$ and go to Step 5 (In practice, such a point is obtained performing Step 3 of this algorithm (Extrapolation).)

else go to Step 4 (Backtracking).

Step 3. *Extrapolation*

Step 3.1 If $(\alpha < \alpha_{\max}$ and $N\alpha > \alpha_{\max})$ then set $\alpha_{trial} \leftarrow \alpha_{\max}$

else set $\alpha_{trial} \leftarrow N\alpha$.

Step 3.2 If $(\alpha \geq \alpha_{\max}$ and $\|P(x_k + \alpha_{trial} d_k) - P(x_k + \alpha d_k)\|_\infty < \max(\epsilon_{abs}, \epsilon_{rel}\|P(x_k + \alpha d_k)\|_\infty))$ then

take $\alpha_k = \alpha$, $x_{k+1} = P(x_k + \alpha d_k)$ and terminate the execution of Algorithm 2.1.

Step 3.3

$$\text{If } (f(P(x_k + \alpha_{trial} d_k)) \geq f(P(x_k + \alpha d_k))) \text{ then} \tag{5}$$

take $\alpha_k = \alpha$, $x_{k+1} = P(x_k + \alpha d_k)$ and go to Step 5

else set $\alpha \leftarrow \alpha_{trial}$ and go to Step 3.1.

Step 4. *Backtracking*

Step 4.1 Compute $\alpha_{new} \in [\sigma_1 \alpha, \sigma_2 \alpha]$ and set $\alpha \leftarrow \alpha_{new}$.

Step 4.2

$$\text{If } (f(x_k + \alpha d_k) \leq f(x_k) + \gamma\alpha\langle g_k, d_k\rangle) \text{ then} \tag{6}$$

take $\alpha_k = \alpha$, $x_{k+1} = x_k + \alpha d_k$ and go to Step 5

else go to Step 4.1.

Step 5. If $\alpha_k \geq \alpha_{max}$ terminate the execution of Algorithm 2.1

else set $k \leftarrow k+1$ and go to Step 1.

Remarks. Let us explain here the main steps of Algorithm 2.1 and their motivations. The algorithm perform line-searches along directions that satisfy the angle-cosine condition (2). In general, this line search will be used with directions that possess some second-order information, so that the "natural" step $\alpha = 1$ must be initially tested and accepted if sufficient-descent and directional-derivative conditions ((3) and (4)) are satisfied.

The first test, at Step 2.1, asks whether $x_k + d_k$ is interior to the box. If this is not the case, but $f(x_k + \alpha_{max} d_k) < f(x_k)$, we try to obtain smaller functional values multiplying the step by a fixed factor and projecting onto the box. This procedure is called "Extrapolation". If $x_k + d_k$ is not interior and $f(x_k + \alpha_{max} d_k) \geq f(x_k)$ we do backtracking.

If $x_k + d_k$ is interior but the Armijo condition (3) does not hold, we also do backtracking. Backtracking stops when the Armijo condition (6) is fulfilled. If (3) holds, we test the directional derivative condition (4). As we mentioned above, if (4) is satisfied too, we accept $x_k + d_k$ as new point. However, if (3) holds and (4) does not, we judge that, very likely, taking larger steps along the direction d_k will produce further decrease of the objective function. So, in this case we also do Extrapolation.

In the Extrapolation procedure we try successive projections of $x_k + \alpha d_k$ onto the box, with increasing values of α. If the entry point $x_k + \alpha d_k$ is interior but $x_k + N\alpha d_k$ is not, we make sure that the point $x_k + \alpha_{max} d_k$ will be tested first. The extrapolation finishes when decrease of the function is not obtained anymore or when the distance between two consecutive projected trial points is negligible.

The iteration of Algorithm 2.1 finishes at Step 5. If the corresponding iterate x_{k+1} is on the boundary of $\mathcal{B}$, the algorithm stops, having encountered a boundary point where the functional value decreased with respect to all previous ones. If x_{k+1} is in the interior of $\mathcal{B}$ the execution continues increasing the iteration number.

The flux-diagrams in Figures 1 and 2 help to understand the structure of the line-search procedure.

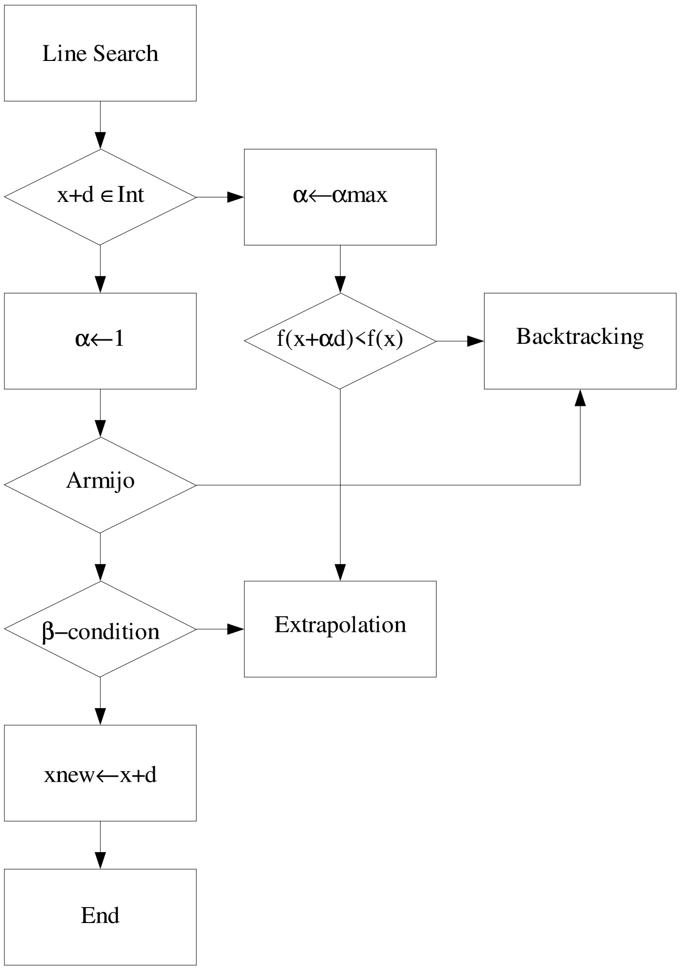

Figure 1: Line Search procedure.

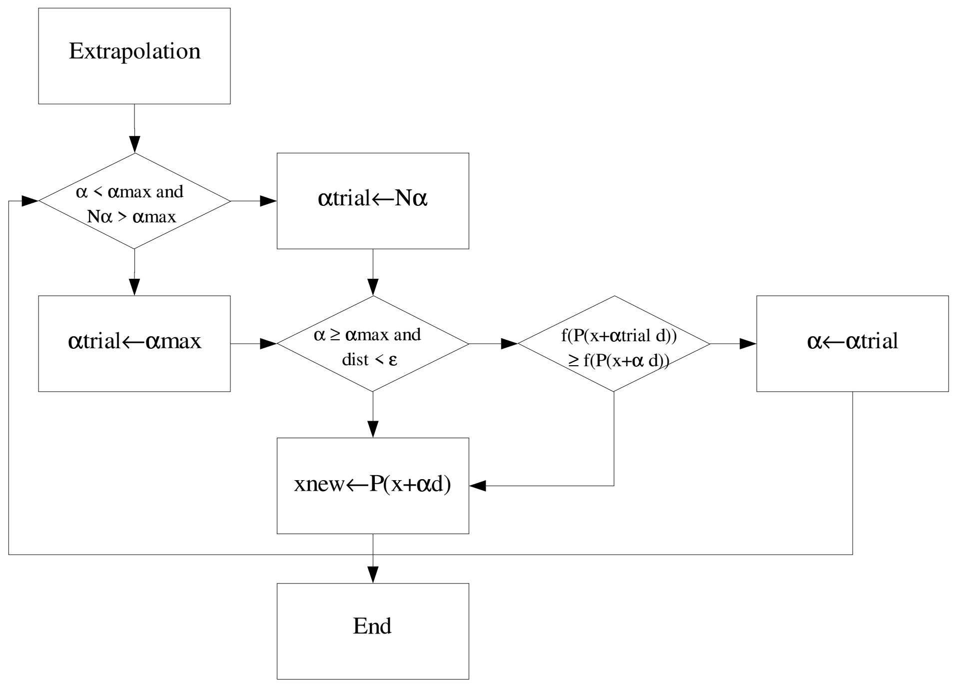

Figure 2: Extrapolation strategy.

In the following theorem we prove that any sequence generated by Algorithm 2.1, either stops at an unconstrained stationary point, or stops in the boundary of $\mathcal{B}$, or generates, in the limit, unconstrained stationary points.

Theorem 2.1. *Algorithm 2.1 is well defined and generates points with strictly decreasing functional values. If $\{x_k\}$ is a sequence generated by Algorithm 2.1, one of the following possibilities holds.*

(i) The sequence stops at x_k, with $g(x_k) = 0$.

(ii) The sequence stops at $x_k \in \mathcal{B} - Int(\mathcal{B})$ and $f(x_k) < f(x_{k-1}) < \ldots < f(x_0)$.

(iii) The sequence is infinite, it has at least one limit point, and every limit point x_* satisfies $g(x_*) = 0$.

Proof. Let us prove first that the algorithm is well defined and that it generates a sequence with strictly decreasing function values. To see that it is well defined we prove that the loops of Steps 3 and 4 necessarily finish in finite time. In fact, at Step 3 we multiply the nonnull direction d_k by a number greater than one, or we take the maximum allowable feasible step. Therefore, eventually, the boundary is reached or the increase condition (5) is met. The loop of Step 4 is a classical backtracking loop and finishes because of well-known directional derivative arguments. See [14]. On exit, the algorithm always requires that $f(x_k + \alpha_k d_k) < f(x_k)$, so the sequence $\{f(x_k)\}$ is strictly decreasing.

It remains to prove that, if neither (i) nor (ii) hold, then any cluster point x_* of the generated sequence satisfies $g(x_*) = 0$. Let K_1 be an infinite subset of $I\!N$ such that

$$\lim_{k \in K_1} x_k = x_*.$$

Define $s_k = x_{k+1} - x_k$ for all $k \in I\!N$. Suppose first that $\|s_k\|$ is bounded away from zero for $k \in K_1$. Therefore, there exists $\rho > 0$ such that $\|s_k\| \geq \rho$ for all $k \in K_1$. By (3) or (6) we have that

$$f(x_{k+1}) \leq f(x_k) + \gamma \langle g_k, s_k \rangle$$

for all $k \in K_1$. Therefore, by (2),

$$f(x_{k+1}) \leq f(x_k) - \gamma\theta \|g_k\| \|s_k\| \leq f(x_k) - \gamma\theta\rho \|g_k\|.$$

By the continuity of f this implies that $\lim_{k\in K_1} \|g_k\| = 0$ and, so, $g(x_*) = 0$.

Suppose now that $\|s_k\|$ is not bounded away from zero for $k \in K_1$. So, there exists K_2, an infinite subset of K_1, such that $\lim_{k\in K_2} \|s_k\| = 0$. Let $K_3 \subset K_2$ be the set of indices such that α_k is computed at Step 2.2 for all $k \in K_3$. Analogously, let $K_4 \subset K_2$ be the set of indices such that α_k is computed at Step 3 for all $k \in K_4$ and let $K_5 \subset K_2$ be the set of indices such that α_k is computed at Step 4 for all $k \in K_5$. We consider three possibilities:

(i) K_3 is infinite.

(ii) K_4 is infinite.

(iii) K_5 is infinite.

Consider, first, the case (i). By (4) we have that

$$\langle g(x_k + s_k), s_k \rangle \geq \beta \langle g_k, s_k \rangle$$

and

$$\langle g(x_k + s_k), \frac{s_k}{\|s_k\|} \rangle \geq \beta \langle g_k, \frac{s_k}{\|s_k\|} \rangle \tag{7}$$

for all $k \in K_3$. Since K_3 is infinite, taking an convergent subsequence $s_k/\|s_k\| \to d$, taking limits in (7) and using continuity, we obtain that

$$\langle g(x_*), d \rangle \geq \beta \langle g(x_*), d \rangle. \tag{8}$$

Since $\beta \in (0, 1)$, this implies that $\langle g(x_*), d \rangle \geq 0$. But, by (2) and continuity, $\langle g(x_*), d \rangle \leq -\theta \|g(x_*)\|$. So, $g(x_*) = 0$.

Consider, now, Case (iii). In this case, K_5 is infinite. For all $k \in K_5$ there exists s'_k such that

$$f(x_k + s'_k) \geq f(x_k) + \gamma \langle g_k, s'_k \rangle \tag{9}$$

and

$$\|s'_k\| \leq \|s_k\|/\sigma_1. \tag{10}$$

By (10), $\lim \|s'_k\| = 0$. Now, by (9), we have for all $k \in K_5$,

$$f(x_k + s'_k) - f(x_k) \geq \gamma \langle g_k, s'_k \rangle.$$

So, by the Mean-Value theorem, there exists $\xi_k \in [0,1]$ such that

$$\langle g(x_k + \xi_k s'_k), s'_k \rangle \geq \gamma \langle g_k, s'_k \rangle$$

for all $k \in K_5$. Dividing by $\|s'_k\|$, taking limits for a convergent subsequence ($s'_k/\|s'_k\| \to d$) we obtain that

$$\langle g(x_*), d \rangle \geq \beta \langle g(x_*), d \rangle.$$

This inequality is similar to (8). So, $g(x_*) = 0$ follows from the same arguments.

Consider, now, Case (ii). Since we are considering cases where an infinite sequence is generated it turns out that, in (5), $P(x_k + \alpha d_k) = x_k + \alpha d_k$. Moreover, by Step 3.1, $\alpha_{trial} \leq N\alpha$ and $P(x_k + \alpha_{trial} d_k) = x_k + \alpha_{trial} d_k$. Therefore, for all $k \in K_4$, writing $\alpha'_k = \alpha_{trial}$, we have that $\alpha'_k \in (\alpha_k, N\alpha_k]$ and

$$f(x_k + \alpha'_k d_k) \geq f(x_k + \alpha_k d_k). \tag{11}$$

Therefore, by the Mean-Value theorem, for all $k \in K_4$ there exists $\xi_k \in [\alpha_k, \alpha'_k]$ such that

$$\langle g(x_k + \xi_k d_k), (\alpha'_k - \alpha_k) d_k \rangle \geq 0.$$

Thus, for all $k \in K_4$, since $\alpha'_k > \alpha_k$, we have that

$$\langle g(x_k + \xi_k d_k), d_k \rangle \geq 0.$$

Since $\|\alpha_k d_k\| \to 0$ we also have that $\|\alpha'_k d_k\| \to 0$ and $\|\xi_k d_k\| \to 0$. So, dividing by $\|d_k\|$ and taking a convergent subsequence of $d_k/\|d_k\|$, we obtain:

$$\langle g(x_*), d \rangle \geq 0.$$

But, by (2), taking limits we get $\langle g(x_*), d \rangle \leq -\theta \|g(x_*)\|$. This implies that $-\theta \|g(x_*)\| \geq 0$. So, $g(x_*) = 0$. This completes the proof. $\square$

3 The box-constrained algorithm

The problem considered in this section is

$$\text{Minimize } f(x) \text{ subject to } x \in \Omega, \tag{12}$$

where Ω is given by (1).

As in [23], let us divide the feasible set Ω into disjoint open faces, as follows. For all $I \subset \{1, 2, \ldots, n, n+1, n+2, \ldots, 2n\}$, we define

$$F_I = \{x \in \Omega \mid x_i = \ell_i \text{ if } i \in I,\ x_i = u_i \text{ if } n+i \in I,\ \ell_i < x_i < u_i \text{ otherwise}\}.$$

We also define V_I the smallest affine subspace that contains F_I and S_I the parallel linear subspace to V_I. The (continuous) projected gradient at $x \in \Omega$ is defined as

$$g_P(x) = P_\Omega(x - g(x)) - x.$$

For all $x \in F_I$, we define

$$g_I(x) = P_{S_I}[g_P(x)].$$

The main algorithm considered in this paper is described below.

Algorithm 3.1: GENCAN

Assume that $x_0 \in \Omega$ is an arbitrary initial point, $\eta \in (0, 1)$ and $0 < \sigma_{min} \leq \sigma_{max} < \infty$. Let F_I be the face that contains the current iterate x_k. Assume that $g_P(x_k) \neq 0$ (otherwise the algorithm terminates). At the main iteration of the algorithm we perform the test

$$\|g_I(x_k)\| \geq \eta \|g_P(x_k)\|. \tag{13}$$

If (13) takes place, we judge that it is convenient that the new iterate belongs to $\bar{F}_I$ (the closure of F_I) and, so, we compute x_{k+1} doing one iteration of Algorithm 2.1, with the set of variables restricted to the free variables in F_I. So, the set $\mathcal{B}$ of the previous section corresponds to $\bar{F}_I$ here.

If (13) does not hold, we decide that some constraints should be abandoned and, so, the new iterate x_{k+1} is computed doing one iteration of the SPG method described by Algorithm 3.2. In this case, before the computation of x_{k+1} we compute the spectral gradient coefficient σ_k in the following way. If $k = 0$ or $\langle (x_k - x_{k-1}), (g(x_k) - g(x_{k-1})\rangle \leq 0$ then

$$\sigma_k = \max\{1, \|x_k\| / \|g_P(x_k)\|\}.$$

Otherwise, define

$$\sigma'_k = \frac{\langle (x_k - x_{k-1}), (g(x_k) - g(x_{k-1})) \rangle}{\|x_k - x_{k-1}\|^2}$$

and

$$\sigma_k = \min\{\sigma_{\max}, \max\{\sigma_{\min}, \sigma'_k\}\}.$$

Algorithm 3.2 is the algorithm used when it is necessary to leave the current face, according to the test (13).

Algorithm 3.2: SPG

Compute x_{k+1} as the next iterate of a monotone SPG iteration [7, 8] with the spectral step σ_k. Namely, we define the search direction d_k as

$$d_k = P_\Omega(x_k - \sigma_k g(x_k)) - x_k$$

and we compute $x_{k+1} = x_k + \alpha_k d_k$ in such a way that

$$f(x_{k+1}) \leq f(x_k) + \gamma \alpha_k \langle g(x_k), d_k \rangle,$$

trying first $\alpha_k = 1$ and, perhaps, reducing this coefficient by means of a safeguarded quadratic interpolation procedure.

Remark. Observe that $x_{k+1} \notin \bar{F}_I$ if $x_k \in F_I$ and x_{k+1} is computed by Algorithm 3.2. In this case, (13) does not hold, so $\|g_P(x_k)\| > \|g_I(x_k)\|$. Since the components corresponding to the free variables of $g_I(x_k)$ and $g_P(x_k)$ are the same, this means that $g_P(x_k)$ has nonnull components corresponding to fixed variables. Therefore, $x_k + \alpha g_P(x_k) \notin \bar{F}_I$ for all $\alpha > 0$. So, $P_\Omega(x_k + \alpha g_P(x_k)) \notin \bar{F}_I$ for all $\alpha > 0$. But, according to the SPG iteration,

$$x_{k+1} = x_k + \alpha' [P_\Omega(x_k + \alpha g_P(x_k)) - x_k]$$

for some $\alpha > 0$, $\alpha' > 0$. This implies that $x_{k+1} \notin \bar{F}_I$.

We finish this section giving some theoretical results. Roughly speaking, we prove that the algorithm is well defined and that a Karush-Kuhn-Tucker point is computed up to an arbitrary precision. Moreover, under dual-nondegeneracy, the (infinite) algorithm identifies the face to which the limit belongs in a finite number of iterations.

Theorem 3.1. *Algorithm 3.1 is well defined.*

Proof. This is a trivial consequence of the fact that Algorithm 2.1 and Algorithm 3.2 (the SPG algorithm [7]) are well defined. □

Theorem 3.2. *Assume that $\{x_k\}$ is generated by Algorithm 3.1. Suppose that there exists $\bar{k} \in \{0, 1, 2, \ldots\}$ such that $x_k \in F_I$ for all $k \geq \bar{k}$. Then, every limit point of $\{x_k\}$ is first-order stationary.*

Proof. In this case, x_{k+1} is computed by Algorithm 2.1 for all $k \geq \bar{k}$. Thus, by Theorem 2.1, the gradient with respect to the free variables tends to zero. By a straightforward projection argument, it follows that $\|g_I(x_k)\| \to 0$. Since (13) holds, this implies that $\|g_P(x_k)\| \to 0$. So, every limit point is first-order stationary. □

Theorem 3.3. *Suppose that for all $k \in \{0, 1, 2, \ldots\}$, $x_k \in F_I$, there exists $k' > k$ such that $x_{k'} \notin F_I$. Then, there exists a limit point of $\{x_k\}$ that is first-order stationary.*

Proof. See Theorem 3.3 of [6]. □

Theorem 3.4. *Suppose that all the stationary points of (12) are nondegenerate. ($\frac{\partial f}{\partial x_i}(x) = 0$ only if $\ell_i < x_i < u_i$.) Then, the hypothesis of Theorem 3.2 (and, hence, its thesis) must hold.*

Proof. See Theorem 3.4 of [6]. □

Theorem 3.5. *Suppose that $\{x_k\}$ is a sequence generated by Algorithm 3.1 and let ε be an arbitrary positive number. Then, there exists $k \in \{0, 1, 2, \ldots\}$ such that $\|g_P(x_k)\| \leq \varepsilon$.*

Proof. This result is a direct consequence of Theorems 3.2 and 3.3. $\square$

4 Implementation

At iteration k of Algorithm 2.1 the current iterate is x_k and we are looking for a direction d_k satisfying condition (2). We use a truncated-Newton approach to compute this direction. To solve the Newtonian system we call Algorithm 4.1 (described below) with $A \approx \nabla^2 f(x_k)$, $b = g(x_k)$, $\bar{l} = l - x_k$, $\bar{u} = u - x_k$, and $\Delta = \max(\Delta_{\min}, 10\|x_k - x_{k-1}\|)$ ($\Delta = \max(\Delta_{\min}, 0.1\|x_0\|)$, if $k = 0$).

The following algorithm applies to the problem

$$\min q(s) = \frac{1}{2} s^T A s + b^T s. \tag{14}$$

The initial approximation to the solution of (14) is $s_0 = 0$. The algorithm finds a point s^* which is a solution or satisfies $q(s^*) < q(s_0)$. Perhaps, the final point is on the boundary of the region defined by $\|s\| \leq \Delta$ and $\bar{l} \leq s \leq \bar{u}$.

Algorithm 4.1: Conjugate gradients

The parameters $\bar{\epsilon} << 1$ and $k_{\max} \in I\!N$ are given. The algorithm starts with $k \leftarrow 0$, s_0 given, $r_0 = As_0 - b$ and $\rho_0 = \|r_0\|^2$.

Step 1. *Test stopping criteria*

If ($\sqrt{\rho_k} \leq \bar{\epsilon}\|b\|$ or $k \geq k_{\max}$) then
set $s^* = s_k$ and terminate.

Step 2. *Compute conjugate gradient direction*

Step 2.1 If ($k = 0$) then set $p_k = -r_k$
else compute $\beta_k = \rho_k / \rho_{k-1}$ and $p_k = -r_k + \beta_k p_{k-1}$.

Step 2.2 If ($p_k^T r_k > 0$), replace $p_k \leftarrow -p_k$.

Step 3. *Compute step*

Step 3.1 Compute $\alpha_{\max} \leftarrow \max\{\alpha \geq 0 \mid s_k + \alpha p_k \in S\}$, where
$S = \{x \in I\!R^{\bar{n}} \mid \|x\| \leq \Delta \text{ and } \bar{l} \leq x \leq \bar{u}\}$.

Step 3.2 Compute $w_k = Ap_k$ and $\gamma_k = p_k^T w_k$.

Step 3.3 If ($\gamma_k > 0$) then set $\alpha_k = \min(\alpha_{\max}, \rho_k / \gamma_k)$.
If ($\gamma_k \leq 0$ and $k = 0$) then set $\alpha_k = \alpha_{\max}$.

If ($\gamma_k \leq 0$ and $k > 0$) then set $\alpha_k = 0$, $s^* = s_k$ and terminate.

Step 4. *Compute new iterate*

Step 4.1 Compute $s_{k+1} = s_k + \alpha_k p_k$.

Step 4.2 If ($b^T s_{k+1} > -\theta \|b\| \|s_{k+1}\|$) then
set $s^* = s_k$ and terminate.

Step 4.3 If ($\alpha_k = \alpha_{\max}$) then
set $s^* = s_{k+1}$ and terminate.

Step 5. Compute $r_{k+1} = r_k + \alpha_k w$, $\rho_{k+1} = \|r_{k+1}\|^2$,
set $k \leftarrow k+1$ and go to Step 1.

This algorithm is a modification of the one presented in [27] (p. 529) for symmetric positive definite matrices A and without constraints. The modifications are the following:

- At Step 2.2 we test if p_k is a descent direction at s_k, i.e., if $\langle p_k, \nabla q(s_k) \rangle < 0$. To force this condition we multiply p_k by -1 if necessary. If the matrix-vector products are computed exactly, this safeguard is not necessary. However, in many cases the matrix-vector product Ap_k is replaced by a finite-difference approximation. For this reason, we perform the test in order to guarantee that the quadratic decreases along the direction p_k.

- At Step 3.3 we test if $p_k^T A p_k > 0$. If this inequality holds, the step α_k in the direction p_k is computed as the minimum among the conjugate-gradient step and the maximum positive step preserving feasibility. If $p_k^T A p_k \leq 0$ and we are at the first iteration of CG, we set $\alpha_k \leftarrow \alpha_{\max}$. In this way CG will stop with $s^* = -\alpha_{\max} g(x_k)$ which surely satisfies the angle condition of Step 4.2. If we are not at iteration zero of CG, we keep the current approximation to the solution of (14) obtained so far.

- At Step 4.2 we test whether the angle condition (2) is satisfied by the new iterate or not. If this condition is not fulfilled, we stop the algorithm with the previous iterate. We also stop the algorithm if the boundary of the feasible set is achieved (Step 4.3).

The convergence criterion $\bar{\epsilon}$ for the conjugate-gradient algorithm is dynamic. It varies linearly with the logarithm of the norm of the continuous projected gradient, beginning with the value $\bar{\epsilon}_i$ and finishing with $\bar{\epsilon}_f$. We define

$$\bar{\epsilon} = \sqrt{10^{a_{\bar{\epsilon}} \log_{10}(\|g_P(x_k)\|_2^2) + b_{\bar{\epsilon}}}},$$

where

$$a_{\bar{\epsilon}} = \frac{\log_{10}(\bar{\epsilon}_f^2 / \bar{\epsilon}_i^2)}{\log_{10}(\epsilon^2 / \|g_P(x_0)\|_2^2)},$$

$$b_{\bar{\epsilon}} = \log_{10}(\bar{\epsilon}_i^2) - a_{\bar{\epsilon}} \log_{10}(\|g_P(x_0)\|_2^2),$$

and ϵ is used in the stopping criterion $\|g_P(x)\|_\infty < \epsilon$ of Algorithm 3.1.

The parameter k_{max} is the maximum number of CG-iterations for each call of the conjugate-gradient algorithm. It also varies dynamically in such a way that more iterations are allowed at the end of the process than at the beginning. The reason is that we want to invest a larger effort in solving quadratic subproblems when we are close to the solution than when we are far from it. In fact,

$$k_{\max} = (1 - \kappa) \max(1, 10 \log_{10}(\bar{n})) + \kappa \bar{n},$$

where

$$\kappa = \min(0, \frac{\log_{10}(\|g_P(x_k)\|_2^2 / \|g_P(x_0)\|_2^2)}{\log_{10}(\epsilon / \|g_P(x_0)\|_2^2)}).$$

In the Incremental-quotient version of GENCAN, $\nabla^2 f(x_k)$ is not computed and the matrix-vector products $\nabla^2 f(x_k)y$ are approximated by

$$\nabla^2 f(x_k) y \approx \frac{g(x_k + ty) - g(x_k)}{t} \tag{15}$$

with $t = \max(\epsilon_{abs}, \epsilon_{rel}\|x_k\|_\infty)/\|y\|_\infty$. In fact, only the components correspondent to free variables are computed and the existence of fixed variables is conveniently exploited in (15).

5 Numerical experiments with the CUTE collection

In order to assess the reliability GENCAN, we tested this method against some well-known alternative algorithms using all the non-quadratic bound-constrained problems with more than 50 variables from the CUTE [12] collection. The algorithms that we used for comparing GENCAN are BOX-QUACAN [26] (see, also, [28]), LANCELOT [11, 12] and the Spectral Projected Gradient method (SPG) (described as SPG2 in [7]; see also [8]). All the methods used the convergence criterion $\|g_P(x)\|_\infty < 10^{-5}$. Any other stopping criteria were inhibited.

In GENCAN we used $\theta = 10^{-6}$, $\gamma = 10^{-4}$, $\beta = 0.5$, $N = 2$, $\sigma_1 = 0.1$, $\sigma_2 = 0.9$ and $\Delta_{\min} = 0.1$ (for Algorithm 2.1), $\eta = 0.1$, $\sigma_{min} = 10^{-10}$, $\sigma_{max} = 10^{10}$ and $\epsilon = 10^{-5}$ (for Algorithm 3.1), and $\bar{\epsilon}_i = 0.1$, $\bar{\epsilon}_f = 10^{-5}$ (for Algorithm 4.1). In all algorithms we used $\epsilon_{rel} = 10^{-7}$ and $\epsilon_{abs} = 10^{-10}$. The parameters of (the line search of) Algorithm 3.2 were the default parameters mentioned in [7] and the same used in (the line search of) Algorithm 2.1, i.e., $\gamma = 10^{-4}$, $\sigma_1 = 0.1$ and $\sigma_2 = 0.9$.

In LANCELOT we used exact second derivatives and we did not use preconditioning in the conjugate-gradient method. The reason for this is that, in GENCAN, the conjugate gradient method for computing directions is also used without preconditioning. The other options of LANCELOT were the default ones. A small number of modifications were made in BOX-QUACAN to provide a fair comparison. These modifications were: (i) the initial trust-region radius of GENCAN was adopted; (ii) the maximum number of conjugate-gradient iterations was fixed in $10 \log_{10}(n)$; (iii) the accuracy for solving the quadratic subproblems was dynamic in BOX-QUACAN varying from 0.1 to 10^{-5}, as done in GENCAN, (iv) the minimum trust-region radius Δ_{min} was fixed in 10^{-3} to be equal to the corresponding parameter in GENCAN.

The codes are written in Fortran 77. The tests were done using an ultraSPARC from SUN, with 4 processors at 167 MHz, 1280 mega bytes of main memory, and Solaris 2.5.1 operating system. The compiler was `WorkShop Compilers 4.2 30 Oct 1996 FORTRAN 77 4.2`. Finally we have used the flag `-O4` to optimize the code.

In the first four tables we report the full performance of LANCELOT, SPG, BOX-QUACAN, GENCAN (true Hessian) and GENCAN (Incremental-quotients). The usual definition of iteration in LANCELOT involves only one function evaluation. However, in order to unify the comparison we call "iteration" to the whole process that computes a new iterate with lower functional value, starting from the current one. Therefore, a single LANCELOT-iteration involves one gradient evaluation but, perhaps, several functional evaluations. At each iteration several trust-region problems are solved approximately and each of them uses a number of CG-iterations. Problems HADAMALS and SCON1LS have bounds where the lower limit is equal to the upper limit. BOX-QUACAN does not run under this circumstances, so the performance of this method in that situation is not reported in the corresponding table. In these tables, we report, for each method:

IT: Number of iterations;

FE: Functional evaluations;

GE: Gradient evaluations;

CG: Conjugate gradient iterations, except in the case of SPG, where CG iterations are not computed;

Time: CPU time in seconds;

$f(x)$**:** final functional value obtained;

$\|g_P(x)\|_\infty$**:** Sup-norm of the projected gradient at the final point.

The next 3 tables repeat the information of the first ones in a more compact and readable form. In Table 5 we report the final functional value obtained for each method, in the cases where there was at least one difference between them, when computed with four significant digits.

In Table 7 we report, for each method, the numbers FE and (GE+CG). The idea is that a CG iteration is sometimes as costly as a gradient-evaluation. The cost is certainly the same when we use the incremental-quotient version of GENCAN. Roughly speaking, GE+CG represents the amount of work used for solving subproblems and FE represents the work done on the true problem trying to reduce the objective function.

Table 8 reports the computer times for the problems where at least one of the methods used more than 1 second. The computer time used by LANCELOT must be considered under the warning made in [9] page 136, *"LANCELOT [...] does not require an interface using the CUTE tools. It is worth noting that LANCELOT exploits much more structure than that*

Problem	n	IT	FE	GE	CG	Time	$f(x)$	$\|g_P(x)\|_\infty$
BDEXP	5000	10	10	11	27	2.13	1.969D−03	6.467D−06
EXPLIN	120	13	13	14	80	0.06	−7.238D+05	3.454D−06
EXPLIN2	120	11	11	12	52	0.05	−7.245D+05	5.428D−06
EXPQUAD	120	15	18	16	98	0.12	−3.626D+06	4.617D−06
MCCORMCK	10000	6	8	7	16	4.24	−9.133D+03	3.841D−06
PROBPENL	500	1	1	2	0	0.07	3.992D−07	3.423D−07
QRTQUAD	120	144	178	145	570	1.39	−3.625D+06	3.505D−06
S368	100	6	8	7	20	2.64	−1.337D+02	1.898D−08
HADAMALS	1024	10	10	11	68	4.40	3.107D+04	1.126D−06
CHEBYQAD	50	22	28	23	463	2.22	5.386D−03	7.229D−06
HS110	50	1	1	2	0	0.01	−9.990D+09	0.000D+00
LINVERSE	1999	22	28	23	2049	47.22	6.810D+02	3.003D−06
NONSCOMP	10000	8	8	9	78	6.30	2.005D−12	5.253D−06
DECONVB	61	13	15	14	246	0.30	6.393D−09	3.596D−06
QR3DLS	610	281	323	282	23614	439.31	2.245D−08	6.136D−06
SCON1LS	1002	8005	9340	8006	5742462	26761.29	5.981D−04	9.720D−06

Table 1: Performance of LANCELOT.

provided by the interface tools". As a consequence, although GENCAN used less iterations, less functional evaluations, less gradient evaluations and less conjugate-gradient iterations than LANCELOT in SCON1LS, its computer time is greater than the one spent by LANCELOT. In some problems, like QR3DLS and CHEBYQAD, the way in which LANCELOT takes advantage of the SIF structure is also impressive.

Now we include an additional table that was motivated by the observation of Table 7. It can be observed that the number of functional evaluations per iteration is larger in GENCAN than in LANCELOT and BOX-QUACAN. The possible reasons are three:

- Many SPG-iterations with, perhaps, many functional evaluations per iteration.
- Many TN-iterations with backtracking.
- Many TN-iterations with extrapolations.

We classify the iterations with extrapolation in successful and unsuccessful ones. A successful extrapolation is an iteration where the extrapolation produced a functional value smaller than the one corresponding to the first

Problem	n	IT	FE	GE	Time	$f(x)$	$\|g_P(x)\|_\infty$
BDEXP	5000	12	13	13	0.53	2.744D−03	7.896D−06
EXPLIN	120	54	57	55	0.01	−7.238D+05	4.482D−06
EXPLIN2	120	56	59	57	0.01	−7.245D+05	5.633D−06
EXPQUAD	120	92	110	93	0.03	−3.626D+06	7.644D−06
MCCORMCK	10000	16	17	17	2.27	−9.133D+03	4.812D−06
PROBPENL	500	2	6	3	0.02	3.992D−07	1.022D−07
QRTQUAD	120	598	1025	599	0.20	−3.624D+06	8.049D−06
S368	100	16	19	17	1.45	−1.403D+02	1.963D−08
HADAMALS	1024	30	42	31	1.63	3.107D+04	2.249D−07
CHEBYQAD	50	841	1340	842	33.75	5.386D−03	9.549D−06
HS110	50	1	2	2	0.00	−9.990D+09	0.000D+00
LINVERSE	1999	1022	1853	1023	33.82	6.810D+02	8.206D−06
NONSCOMP	10000	43	44	44	2.81	3.419D−10	7.191D−06
DECONVB	61	1670	2560	1671	1.71	4.826D−08	9.652D−06
QR3DLS	610	105918	228722	105919	2203.97	1.973D−05	9.986D−06
SCON1LS	1002	5000001	7673022	5000002	116189.70	1.224D+00	8.578D−05

Table 2: Performance of SPG.

trial point. An unsuccessful extrapolation corresponds to a failure in the first attempt to “double” the steplength. Therefore, in an unsuccessful extrapolation, an additional “unnecessary” functional evaluation is done and the “next iterate” corresponds to the first trial point. According to this, we report, in Table 9, the following features of GENCAN (incremental-quotient version):

- SPG-IT: SPG iterations, used for leaving the current face.
- SPG-FE: functional evaluations in SPG-iterations.
- TN-IT: TN iterations.
- TN-FE: functional evaluations in TN-iterations.
- TN-(Step 1)-IT: TN-iterations where the unitary step was accepted.
- TN-(Step 1)-FE: functional evaluations in TN-iterations where the unitary step was accepted. This is necessarily equal to TN-(Step 1)-IT.
- TN-(Backtracking)-IT: TN-iterations where backtracking was necessary.

Problem	n	IT	FE	GE	CG	Time	$f(x)$	$\|g_P(x)\|_\infty$
BDEXP	5000	10	11	11	20	1.23	1.967D−03	5.742D−06
EXPLIN	120	16	17	17	96	0.03	−7.238D+05	2.302D−07
EXPLIN2	120	13	14	14	70	0.03	−7.245D+05	1.090D−06
EXPQUAD	120	18	21	19	159	0.09	−3.626D+06	4.024D−06
MCCORMCK	10000	8	9	9	28	5.23	−9.133D+03	6.388D−07
PROBPENL	500	2	3	3	4	0.03	3.992D−07	1.994D−07
QRTQUAD	120	22	28	23	214	0.10	−3.625D+06	5.706D−07
S368	100	7	9	8	73	9.16	−1.402D+02	2.432D−08
CHEBYQAD	50	52	66	53	960	45.70	5.387D−03	9.535D−06
HS110	50	1	2	2	3	0.01	−9.990D+09	0.000D−00
LINVERSE	1999	13	19	14	402	18.15	6.810D+02	5.273D−06
NONSCOMP	10000	11	12	12	69	7.11	1.279D−13	1.461D−06
DECONVB	61	17	21	18	413	0.63	5.664D−03	6.559D−06
QR3DLS	610	3087	3108	3088	70433	2286.09	1.450D−05	7.685D−06

Table 3: Performance of BOX-QUACAN.

- TN-(Backtracking)-FE: functional evaluations at iterations with backtracking.
- TN-(Extrap(+))-IT: successful iterations with extrapolation.
- TN-(Extrap(+))-FE: functional evaluations at successful iterations with extrapolation.
- TN-(Extrap(−))-IT: unsuccessful iterations with extrapolation.
- TN-(Extrap(−))-FE: functional evaluations at unsuccessful iterations with extrapolation. This number is necessarily equal to twice the corresponding number of iterations.

Problem	n	IT	FE	GE	CG	Time	$f(x)$	$\|g_P(x)\|_\infty$
BDEXP	5000	1	12	3	1	0.35	0.000D+00	0.000D+00
EXPLIN	120	17	43	19	39	0.02	−7.238D+05	5.505D−06
EXPLIN2	120	15	45	16	27	0.01	−7.245D+05	3.567D−07
EXPQUAD	120	21	51	23	54	0.04	−3.626D+06	1.086D−06
MCCORMCK	10000	5	18	7	19	4.57	−9.133D+03	1.241D−09
PROBPENL	500	2	7	4	2	0.03	3.992D−07	2.104D−07
QRTQUAD	120	29	73	32	73	0.06	−3.625D+06	4.008D−06
S368	100	9	37	10	14	3.10	−1.360D+02	1.526D−11
HADAMALS	1024	10	18	13	10	1.80	3.107D+04	4.759D−10
CHEBYQAD	50	22	40	23	472	13.86	5.386D−03	3.813D−06
HS110	50	1	3	3	1	0.00	−9.990D+09	0.000D+00
LINVERSE	1999	14	34	16	71	3.88	6.820D+02	1.052D−06
NONSCOMP	10000	17	43	19	32	4.81	7.642D−11	1.161D−06
DECONVB	61	61	138	62	875	1.29	6.043D−11	6.811D−06
QR3DLS	610	305	452	306	27503	976.13	1.960D−10	2.992D−06
SCON1LS	1002	4742	5913	4804	3012141	55779.87	1.269D−03	9.053D−06

Table 4: Performance of GENCAN (true-hessian version).

Problem	n	IT	FE	GE	CG	Time	$f(x)$	$\|g_P(x)\|_\infty$
BDEXP	5000	1	12	3	1	0.36	0.000D+00	0.000D+00
EXPLIN	120	17	43	19	39	0.01	−7.238D+05	5.483D−06
EXPLIN2	120	15	45	16	27	0.01	−7.245D+05	3.562D−07
EXPQUAD	120	21	51	23	53	0.03	−3.626D+06	2.236D−06
MCCORMCK	10000	5	18	7	19	3.56	−9.133D+03	1.241D−09
PROBPENL	500	3	8	5	3	0.03	3.992D−07	1.989D−07
QRTQUAD	120	29	75	33	68	0.04	−3.625D+06	2.296D−06
S368	100	9	37	10	14	2.50	−1.360D+02	1.384D−11
HADAMALS	1024	10	18	13	10	1.19	3.107D+04	4.705D−10
CHEBYQAD	50	31	43	32	886	22.26	5.386D−03	2.929D−06
HS110	50	1	3	3	1	0.00	−9.990D+09	0.000D+00
LINVERSE	1999	14	34	16	71	2.39	6.820D+02	1.052D−06
NONSCOMP	10000	18	55	20	34	4.58	4.728D−18	1.428D−08
DECONVB	61	57	172	58	511	0.43	3.925D−10	9.061D−06
QR3DLS	610	308	476	309	27209	523.50	2.075D−10	1.238D−06
SCON1LS	1002	7283	8565	7352	4987908	73606.98	4.549D−04	9.578D−06

Table 5: Performance of GENCAN (incremental-quotient version).

Problem	LANCELOT	SPG	BOX-QUACAN	GENCAN-HESS	GENCAN-QUOT
BDEXP	1.969D−03	2.744D−03	1.967D−03	0.000D+00	0.000D+00
QRTQUAD	−3.625D+06	−3.624D+06	−3.625D+06	−3.625D+06	−3.625D+06
S368	−1.337D+02	−1.403D+02	−1.402D+02	−1.360D+02	−1.360D+02
CHEBYQAD	5.386D−03	5.386D−03	5.387D−03	5.386D−03	5.386D−03
LINVERSE	6.810D+02	6.810D+02	6.810D+02	6.820D+02	6.820D+02
NONSCOMP	2.005D−12	3.419D−10	1.279D−13	7.642D−11	4.728D−18
DECONVB	6.393D−09	4.826D−08	5.664D−03	6.043D−11	3.925D−10
QR3DLS	2.245D−08	1.973D−05	1.450D−05	1.960D−10	2.075D−10
SCON1LS	5.981D−04	1.224D+00	—	1.269D−03	4.549D−04

Table 6: Final functional values.

Problem	LANCELOT		SPG		BOX-QUACAN		GENCAN-QUOT	
	FE	GE+CG	FE	GE	FE	GE+CG	FE	GE+CG
BDEXP	10	38	13	13	11	30	12	4
EXPLIN	13	94	57	55	17	112	43	58
EXPLIN2	11	64	59	57	14	83	45	43
EXPQUAD	18	114	110	93	21	177	51	76
MCCORMCK	8	23	17	17	9	36	18	26
PROBPENL	1	2	6	3	3	6	8	8
QRTQUAD	178	715	1025	599	28	236	75	101
S368	8	27	19	17	9	80	37	24
HADAMALS	10	79	42	31	—	—	18	23
CHEBYQAD	28	486	1340	842	66	1012	43	918
HS110	1	2	2	2	2	4	3	4
LINVERSE	28	2072	1853	1023	19	415	34	87
NONSCOMP	8	87	44	44	12	80	55	54
DECONVB	15	260	2560	1671	21	430	172	569
QR3DLS	323	23896	228722	105919	3108	73520	476	27518
SCON1LS	9340	5750468	7673022	5000002	—	—	8565	4995260

Table 7: Functional and equivalent-gradient (GE+CG) evaluations.

Problem	LANCELOT	SPG	BOX-QUACAN	GENCAN-HESS	GENCAN-QUOT
BDEXP	2.13	0.53	1.23	0.35	0.36
MCCORMCK	4.24	2.27	5.23	4.57	3.56
QRTQUAD	1.39	0.20	0.10	0.06	0.04
S368	2.64	1.45	9.16	3.10	2.50
HADAMALS	4.40	1.63	—	1.80	1.19
CHEBYQAD	2.22	33.75	45.70	13.86	22.26
LINVERSE	47.22	33.82	18.15	3.88	2.39
NONSCOMP	6.30	2.81	7.11	4.81	4.58
DECONVB	0.30	1.71	0.63	1.29	0.43
QR3DLS	439.31	2203.97	2286.09	976.13	523.50
SCON1LS	26761.29	116189.70	—	55779.87	73606.98

Table 8: Computer time.

	Type of GENCAN iterations				Details of Truncated Newton iterations							
	SPG Iteration		TN iterations		Step=1		Backtracking		Extrap(+)		Extrap(−)	
Problem	IT	FE	IT	FE	IT	FE	IT	FE	IT	FE	IT	FE
BDEXP	0	0	1	11	0	0	0	0	1	11	0	0
EXPLIN	1	1	16	41	7	7	0	0	5	26	4	8
EXPLIN2	1	1	14	43	4	4	0	0	5	29	5	10
EXPQUAD	2	6	19	44	8	8	3	7	2	17	6	12
MCCORMCK	0	0	5	17	3	3	0	0	1	12	1	2
PROBPENL	0	0	3	7	2	2	0	0	1	5	0	0
QRTQUAD	2	9	27	65	16	16	6	14	4	33	1	2
S368	0	0	9	36	4	4	1	2	4	30	0	0
HADAMALS	0	0	10	17	6	6	1	2	1	5	2	4
CHEBYQAD	0	0	31	42	24	24	7	18	0	0	0	0
HS110	0	0	1	2	0	0	0	0	1	2	0	0
LINVERSE	0	0	14	33	6	6	7	22	1	5	0	0
NONSCOMP	0	0	18	54	2	2	2	14	4	18	10	20
DECONVB	8	58	49	113	14	14	0	0	11	51	24	48
QR3DLS	0	0	308	475	218	218	90	257	0	0	0	0
SCON1LS	0	0	7283	8564	6848	6848	367	1394	63	312	5	10
TOTAL	14	75	7808	9564	7162	7162	484	1730	104	556	58	116

Table 9: GENCAN features.

Observing Table 9 we realise that:

1. The number of SPG-iterations is surprisingly small. Therefore, only in few iterations the mechanism to "leave the face" is activated. So, in most iterations, the number of active constraints remains the same or is increased. Clearly, SPG-iterations are not responsible for the relatively high number of functional evaluations.

2. The number of iterations where backtracking was necessary is, also, surprisingly small. Therefore, extrapolations are responsible for the functional-evaluations phenomenon. Since an unsuccessful extrapolation uses only one additional (unnecessary) functional evaluations, its contribution to increasing FE is also moderate. In fact, unsuccessful extrapolations are responsible for 116 functional evaluations considering all the problems, this means less than 8 evaluations per problem. It turns out that many functional evaluations are used in successful extrapolations. Considering the overall performance of the method this seems to be a really good feature. An extreme case is BDEXP, where only one TN-iteration was performed, giving a successful extrapolation that used 11 functional evaluations and gave the solution of the problem.

Further remarks

Convergence was obtained for all the problems with all the methods tested, with the exception of SPG that did not solve SCON1LS after more than thirty hours of computer time. The method that, in most cases, obtained the lowest functional values was GENCAN-QUOT, but the differences do not seem to be large enough to reveal a clear tendency.

As was already mentioned in [7], the behavior of SPG is surprisingly good. Although it is the only method that fails to solve a problem in reasonable time, its behavior in the problems where it works is quite efficient. This indicates the existence of families of problems where SPG is, probably, the best possible alternative. This observation has already been made in [8].

BOX-QUACAN has been the less successful method in this set of experiments. This is not surprising, since the authors of [16] had observed that this method outperformed LANCELOT in quadratic problems but is not so good when the function is far from being quadratic. In fact, it was this observation what motivated the present work. Nevertheless, there is still a large scope for improvements of BOX-QUACAN, as far as we take into account that improvements in the solution of quadratic subproblems are possible and that sophisticated strategies for updating trust-region radius can be incorporated.

6 Experiments with very large problems

We wish to place q circles of radius r in the rectangle $[0, d_1] \times [0, d_2]$ in such a way that for all $i = 1, \dots q$ and for all $j \in I_i \subset \{1, \dots, q\}$, the intersection between the circle i and the circle j is at most one point. Therefore, given $I_i \subset \{1, \dots, q\}$, $i = 1, \dots, q$, the goal is to determine $c^1, \dots, c^q \in [r, d_1 - r] \times [r, d_2 - r]$ solving the problem:

$$\text{Minimize} \sum_{i=1}^{q} \sum_{j \in I_i} \max(0, 2r - \|c^i - c^j\|_2^2)^2$$

subject to

$$r \leq c_1^i \leq d_1 - r, \text{ and}$$

$$r \leq c_2^i \leq d_2 - r, \text{ for } i = 1, \dots, q.$$

The points $c^1, \dots, c^q$ are the centers of the desired circles. If the objective function value at the solution of this minimization problem is zero, then the original problem is solved.

When $I_i = \{1, \dots, q\} - \{i\}$ for all $i = 1, \dots, q$ the problem above is known as the Cylinder Packing problem [22]. The present generalization is directed to Sociometry applications.

Table 10 describes the main features of some medium and large scale problems of this type. In the problems 9-15 the sets I_i were randomly generated with the Schrage's random number generator [35] and seed = 1. In all cases we used $r = 0.5$. Observe that n, the number of variables, is equal to 2 q.

Tables 11 and 12 show the performances of GENCAN and LANCELOT. Internal limitations of the big-problems installation of CUTE forbid the solution of larger instances of this problems using SIF. We show the CPU

Problem	n	$\#I_i$	box
1	400	199	100×100
2	400	199	75×75
3	400	199	50×50
4	400	199	25×25
5	500	249	100×100
6	500	249	75×75
7	500	249	50×50
8	500	249	25×25
9	10^5	10	25×2
10	5×10^5	10	25×3
11	10^6	10	30×3
12	5×10^6	10	30×4
13	10^7	2	40×4
14	10^7	5	40×4
15	10^7	10	40×5

Table 10: Medium- and large-scale classical and modified cylinder packing problems.

times of GENCAN both using SIF (SIF-Time) and Fortran subroutines (FS-Time) for computing function and gradient. We used a random initial point (generated inside the box with the Schrage's algorithm and seed equal to 1). Both methods found a global solution in all the cases. In Table 12 we also report the number of free variables at the solution so far found by GENCAN.

Problem(n)	GENCAN										LANCELOT				
	using Fortran subroutines					using SIF									
	IT	FE	GE	CG	Time	IT	FE	GE	CG	Time	IT	FE	GE	CG	Time
1(400)	2	3	3	2	0.26	2	3	3	2	0.85	2	3	3	2	1.63
2(400)	2	3	3	2	0.25	2	4	3	2	0.78	2	3	3	1	1.60
3(400)	3	10	4	4	0.51	3	9	4	5	1.44	3	4	4	2	2.00
4(400)	9	17	10	29	1.75	7	16	8	13	3.55	9	8	10	11	5.04
5(500)	2	3	3	3	0.44	1	4	2	1	0.89	2	3	3	1	3.08
6(500)	1	5	2	1	0.33	2	4	3	2	1.57	4	4	5	5	4.43
7(500)	4	8	5	7	0.93	4	10	5	7	3.20	5	5	6	6	5.27
8(500)	9	23	10	19	2.40	7	21	8	15	6.50	49	48	50	22	25.88

Table 11: GENCAN and LANCELOT with cylinder packing problems.

Problem(n)		IT	FE	GE	CG	Time	nfree
9	(10^5)	14	73	15	23	61.66	92718
10	(5×10^5)	26	140	27	81	870.31	485347
11	(10^6)	20	90	21	45	1123.19	960323
12	(5×10^6)	21	118	22	99	9350.60	4889338
13	(10^7)	6	52	7	7	1772.61	9957240
14	(10^7)	9	63	10	18	4560.70	9906040
15	(10^7)	14	84	15	28	10649.79	9914682

Table 12: GENCAN with very large problems.

7 Final remarks

Numerical algorithms must be analyzed not only from the point of view of its present state but also from considerations related to their possibility of improvement. The chances of improvement of active-set methods like the one presented in this paper come from the development of new unconstrained algorithms and from the adaptation of known unconstrained algorithms to the specific characteristics of our problem. In our algorithm, the computation of the search direction is open to many possibilities. As we mentioned in the introduction, a secant multipoint scheme (with a different procedure for leaving the faces) was considered in [10] and a negative-curvature Newtonian direction for small problems was used in [6], where leaving faces is also associated to SPG. A particularly interesting alternative is the preconditioned spectral projected gradient method introduced in [30].

The extension of the technique of this paper to general linearly constrained optimization is another interesting subject of possible research. From the theoretical point of view, the extension is straightforward, and the convergence proofs do not offer technical difficulties. The only real difficulty is that we need to project onto the feasible set, both in the extrapolation steps and in the SPG iterations. In theory, extrapolation can be avoided without affecting global convergence, but projections are essential in SPG iterations. Sometimes, the feasible polytope is such that projections are easy to compute. See [8]. In those cases, the extension of GENCAN would probably be quite efficient.

Acknowledgements.

The authors are very grateful to Nick Gould, who helped them in the use of the SIF language. We are also indebted to two anonymous referees whose comments helped us a lot to improve the final version of the paper.

References

[1] R. Andreani, A. Friedlander and S. A. Santos. On the resolution of the generalized nonlinear complementarity problem. To appear in *SIAM Journal on Optimization*.

[2] R. H. Bielschowsky, A. Friedlander, F. M. Gomes, J. M. Martínez and M. Raydan. An adaptive algorithm for bound constrained quadratic minimization. *Investigación Operativa* 7, pp. 67–102 (1998).

[3] R. H. Byrd, P. Lu, J. Nocedal and C. Zhu, A limited memory algorithm for bound constrained minimization. *SIAM Journal on Scientific Computing* 16, pp. 1190–1208 (1995).

[4] E. G. Birgin, R. Biloti, M. Tygel and L.T. Santos. Restricted optimization: a clue to a fast and accurate implementation of the Common Reflection Surface stack method. *Journal of Applied Geophysics* 42, pp. 143–155 (1999).

[5] E. G. Birgin, I. Chambouleyron and J. M. Martínez. Estimation of the optical constants and the thickness of thin films using unconstrained optimization. *Journal of Computational Physics* 151, pp. 862–880 (1999).

[6] E. G. Birgin and J. M. Martínez. A box constrained optimization algorithm with negative curvature directions and spectral projected gradients. *Computing [Suppl]* 15, pp. 49–60 (2001).

[7] E. G. Birgin, J. M. Martínez and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM Journal on Optimization* 10, pp. 1196–1211 (2000).

[8] E. G. Birgin, J. M. Martínez and M. Raydan. SPG: Software for convex-constrained optimization. *ACM Transactions on Mathematical Software* 27, pp. 340–349 (2001).

[9] I. Bongartz, A. R. Conn, N. I. M. Gould and Ph. L. Toint. CUTE: constrained and unconstrained testing environment. *ACM Transactions on Mathematical Software* 21, pp. 123–160 (1995).

[10] O. Burdakov, J. M. Martínez and E. A. Pilotta. A limited-memory multipoint secant method for bound-constrained optimization. Technical Report, Institute of Mathematics, University of Campinas.

[11] A. R. Conn, N. I. M. Gould and Ph. L. Toint. Global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM Journal on Numerical Analysis* 25, pp. 433–460 (1988).

[12] A. R. Conn, N. I. M. Gould and Ph. L. Toint. A globally convergent augmented Lagrangean algorithm for optimization with general constraints and simple bounds, *SIAM Journal on Numerical Analysis* 28, pp. 545–572 (1991).

[13] A. R. Conn, N. I. M. Gould and Ph. L. Toint. *Trust-Region Methods*. MPS-SIAM Series on Optimization, SIAM, Philadelphia, 2000.

[14] J. E. Dennis and R. B. Schnabel. *Numerical methods for unconstrained optimization and nonlinear equations*. Prentice Hall Series in Computational Mathematics, Englewood Cliffs, New Jersey, 1983.

[15] M. A. Diniz-Ehrhardt, Z. Dostál, M. A. Gomes-Ruggiero, J. M. Martínez and S. A. Santos. Nonmonotone strategy for minimization of quadratics with simple constraints. *Applications of Mathematics* 46, pp. 321–338 (2001).

[16] M. A. Diniz-Ehrhardt, M. A. Gomes-Ruggiero and S. A. Santos. Comparing the numerical performance of two trust-region algorithms for large-scale bound-constrained minimization, in *International Workshop of Numerical Linear Algebra and Optimization*, edited by R. J. B. Sampaio and J. Y. Yuan, Department of Mathematics, Universidade Federal do Paraná, Brazil, pp. 23–24 (1997).

[17] Z. Dostál. Box constrained quadratic programming with proportioning and projections. *SIAM Journal on Optimization* 7, pp. 871–887 (1997).

[18] Z. Dostál, A. Friedlander and S. A. Santos. Solution of coercive and semicoercive contact problems by FETI domain decomposition. *Contemporary Mathematics* 218, pp. 82–93 (1998).

[19] Z. Dostál, A. Friedlander and S. A. Santos. Augmented Lagrangians with adaptive precision control for quadratic programming with equality constraints. *Computational Optimization and Applications* 14, pp.1–17 (1999).

[20] Z. Dostál, A. Friedlander and S. A. Santos. Adaptive Precision Control in Quadratic Programming with Simple Bounds and/or Equalities. To appear in *High Performance Algorithms and Software in Nonlinear Optimization: Applied Optimization*, edited by R. De Leone, A. Murli, P. Pardalos, G. Toraldo, editors, Kluwer.

[21] Z. Dostál, F. A. M. Gomes and S. A. Santos. Solution of contact problems by FETI domain decomposition with natural coarse space projections, *Journal of Computational and Applied Mathematics* 126, pp. 397–415 (2000).

[22] K. Dowsland. Optimising the palletisation of cylinders in cases, *OR Spektrum* 13, pp. 204–212 (1991).

[23] A. Friedlander and J. M. Martínez. On the maximization of a concave quadratic function with box constraints. *SIAM Journal on Optimization* 4, pp. 177–192 (1994).

[24] A. Friedlander, J. M. Martínez, B. Molina and M. Raydan. Gradient methods with retards and generalizations. *SIAM Journal on Numerical Analysis* 36, pp. 275–289 (1999).

[25] A. Friedlander, J. M. Martínez and M. Raydan. A new method for large-scale box constrained convex quadratic minimization problems. *Optimization Methods and Software* 5, pp. 57–74 (1995).

[26] A. Friedlander, J. M. Martínez and S. A. Santos. A new trust region algorithm for bound constrained minimization. *Applied Mathematics and Optimization* 30, pp. 235–266 (1994).

[27] G. H. Golub and C. F. Van Loan. *Matrix Computations.* The Johns Hopkins University press, Baltimore and London, Third Edition, 1996.

[28] N. Krejić, J. M. Martínez, M. P. Mello and E. A. Pilotta. Validation of an augmented Lagrangian algorithm with a Gauss-Newton Hessian approximation using a set of hard-spheres problems. *Computational Optimization and Applications* 16, pp. 247–263 (2000).

[29] C. J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM Journal on Optimization* 9 pp. 1100–1127 (1999).

[30] F. Luengo, M. Raydan, W. Glunt and T.L. Hayden. Preconditioned spectral gradient method for unconstrained optimization problems, Technical Report R.T. 96-08, Computer Science Department, Universidad Central de Venezuela. To appear in *Numerical Algorithms.*

[31] J. M. Martínez. BOX-QUACAN and the implementation of Augmented Lagrangian algorithms for minimization with inequality constraints. *Computational and Applied Mathematics* 19, pp. 31-56 (2000).

[32] M. Mulato, I. Chambouleyron, E. G. Birgin and J. M. Martínez. Determination of thickness and optical constants of a-Si:H films from transmittance data. *Applied Physics Letters* 77, pp. 2133–2135 (2000).

[33] M. Raydan. On the Barzilai and Borwein choice of steplength for the gradient method, *IMA Journal of Numerical Analysis* 13, pp. 321–326 (1993).

[34] M. Raydan. The Barzilai and Borwein gradient method for the large scale unconstrained minimization problem. *SIAM Journal on Optimization* 7, pp. 26–33 (1997).

[35] L. Schrage. A more portable Fortran random number generator. *ACM Transactions on Mathematical Software* 5, pp. 132–138 (1979).

[36] J. Zhang and C. Xu. A class of indefinite dogleg path methods for unconstrained minimization. *SIAM Journal on Optimization* 9, pp. 646–667 (1999).