Practical active-set Euclidian trust-region method with spectral projected gradients for bound-constrained minimization

Marina Andretta * Ernesto G. Birgin † José Mario Martínez ‡

December 1, 2004

Abstract

A practical active-set method for bound-constrained minimization is introduced. Within the current face the classical Euclidian trust-region method is employed. Spectral projected gradient directions are used to abandon faces. Numerical results are presented.

Key words: Bound-constrained optimization, projected gradient, spectral gradient, trust regions.

1 Introduction

The problem of minimizing a continuous function with bounds on the variables has many practical applications.

In recent papers [2, 3, 8] the efficiency of active-set methods has been emphasized. In an active-set method, the domain of the problem is divided into disjoint faces. At each iteration, an appropriate test evaluates whether it is convenient to stay in the current face or to abandon it. When the decision is to stay in the current face an unconstrained minimization method (the "inner algorithm") is applied, regarding only the variables that are free at that face. Iterations of the unconstrained (inner) algorithm are executed as far as a boundary point is not encountered or a leaving-face criterion is not satisfied.

Many different inner algorithms can be used. Essentially, every unconstrained minimization method defines an inner algorithm, but some technical difficulties must be removed. A truncated Newton approach is used in [3]. In [8] the authors use a quasi-Newton method and in [2] an unconstrained algorithm based on negative curvature directions is employed (see [24]). The leaving-face criteria used in those papers has proved to be extremely efficient in

*Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão 1010 Cidade Universitária, 05508-090 São Paulo, SP - Brazil (andretta@ime.usp.br). Sponsored by CNPq.

†Department of Computer Science, Institute of Mathematics and Statistics, University of São Paulo, Rua do Matão 1010 Cidade Universitária, 05508-090 São Paulo, SP - Brazil (egbirgin@ime.usp.br). Sponsored by FAPESP (Grants 03/09169-6 and 01/04597-4), CNPq (PROSUL 490333/2004-4 and grant 300151/00-4) and PRONEX-CNPq/FAPERJ E-26/171.164/2003 - APQ1.

‡Departamento de Matemática Aplicada, IMECC-UNICAMP, CP 6065, 13081-970 Campinas SP, Brazil (martinez@ime.unicamp.br). Sponsored by FAPESP (Grant 01/04597-4), CNPq, FAEP-UNICAMP and PRONEX-CNPq/FAPERJ E-26/171.164/2003 - APQ1.

practice in the sense that, in most cases, the number of leaving-face iterations is small with respect to the total number of iterations. Therefore, the possibility of improving this type of active-set algorithms relies essentially on the choice of the inner method.

In this paper the active-set strategy is based on the use of the classical Euclidian trust-region method within the faces. See [20, 23].

The difficulty in the implementation of the Euclidian trust-region strategy in the presence of bounds is that the intersection between balls and boxes is not a simple set. So, finding the minimizer of a quadratic approximation on such a set is not straightforward and the intersection technique of constrained trust-region algorithms [18, 19] is not easy to apply. It is usually claimed that trust regions based on the sup-norm are better because they fit the geometry of the domain of the problem. The reason is that intersection of boxes are boxes. See [10, 16]. However, to find *global* minimizers of a (not necessarily convex) quadratic on a box is not as simple as to find minimizers of arbitrary quadratics on a ball [17, 20, 21].

Some authors employ ellipsoidal trust regions for defining interior point methods for the box-constrained minimization problem. See [9], the first algorithm of [11] and others. Dennis and Vicente [11] observed that experimental studies on interior point methods show that it is important to allow trial steps to be out of the trust ellipsoid as long as they still satisfy the bound constraints. These authors motivate their "second algorithm" on the fact that there is no reason to use an ellipsoid to define the shape of the trust region if it is not useful for enforcing the bounds. In their practical second algorithm the trust regions are spherical and their experiments show that this algorithm is slightly better than the first.

We go one step further. Even the second "Euclidian trust-region" algorithm of Dennis and Vicente maintains the interiority of all the iterates. Recent research in box-constrained trust-region methods show that, with the help of the spectral projected gradient procedure for escaping from exploited faces, active-set methods can be very effective. Moreover, when we work on a face of low dimension, the linear algebra work associated with the solution of the trust-region subproblem is smaller than when one solves interior trust-region problems. Therefore, the development of an active-set Euclidian trust-region algorithm seems to be attractive although technical difficulties must be removed.

In this paper we define an active-set method that uses Euclidian trust regions inside the faces and where some ad hoc techniques are used in order to obtain theoretical convergence and practical efficiency. The strategy for leaving faces is the one used in [3]. However, within the current face, the pure trust-region algorithm must be modified if the minimizer of a trust-region subproblem is infeasible. Moreover, when the current point is very close to the boundary, solving the trust-region subproblem might not be worthwhile, so a modified iteration must be used. There is a theoretical prize that must be paid for such modifications. First-order convergence is always obtained but second-order convergence is obtained only with respect to points that are not very close to the boundary of the current face. However, we will show in computational experiments how serious is this inconvenient in practice.

For mathematical precedence reasons, we describe the inner algorithm used within the faces in Section 2, whereas in Section 3 we define the box-constrained algorithm (BETRA) and we prove convergence. In spite of this, the reader interested in obtaining a quick overview about the method should consider the possibility of reading first Section 3. In Section 4 numerical experiments are presented. Conclusions are given in Section 5.

2 Inner algorithm

The active-set method uses the general scheme described in [3]. The box will be divided into "open faces". Within each open face, an inner algorithm is used that modifies only free variables. This inner algorithm can be described as an interior-point (or unconstrained) method that, perhaps, hits the boundary of the convex set where the objective function is defined.

Let $A \subset \mathbb{R}^m$ be a closed, bounded and convex set with nonempty interior. (A will be a box in Section 3.) Assume that $\bar{f} : \mathbb{R}^m \to \mathbb{R}$ has continuous third-order derivatives on an open set that contains A. We use the following notation: $\| \cdot \| = \| \cdot \|_2$, $\bar{g} = \nabla \bar{f}$, $\bar{B}(x, \delta) = \{z \in \mathbb{R}^m \mid \|z - x\| \leq \delta\}$. The set of interior points of A will be denoted A^o. $P_C(z)$ will denote the Euclidian projection of z onto the closed and convex set C.

Let us define the typical procedures that are used in the inner algorithm.

Trust-region subproblem

Given $x^k \in A^o$, $\Delta > 0$, $B_k \in \mathbb{R}^{m \times m}$ symmetric, define

$$\psi_k(x) = \frac{1}{2}(x - x^k)^T B_k (x - x^k) + \bar{g}(x^k)^T (x - x^k)$$

and solve

$$\text{Minimize } \psi_k(x) \text{ s.t. } \|x - x^k\| \leq \Delta.$$

Observe that, although x^k is an interior point of A, a solution of the trust-region subproblem might not belong to A.

Inner Spectral Projected Gradient (SPG) iteration [4, 5, 6]

Given $x^k \in A^o$, $0 < \lambda_{\min} < \lambda_{\max} < \infty$, $\lambda^k \in [\lambda_{\min}, \lambda_{\max}]$, $\alpha \in (0, \frac{1}{2})$, and assuming that $\bar{g}(x^k) \neq 0$, perform the following steps:

(a) Compute

$$d^k = P_A(x^k - \lambda^k \bar{g}(x^k)) - x^k.$$

(b) Set $t \leftarrow 1$.

(c) If

$$\bar{f}(x^k + t\, d^k) \leq \bar{f}(x^k) + \alpha\, t\, \bar{g}(x^k)^T d^k \tag{1}$$

set $t_k = t$ and finish the iteration defining $y^k = x^k + t_k d^k$. If (1) does not hold, choose $t_{new} \in [0.1\, t, 0.5\, t]$, set $t \leftarrow t_{new}$ and repeat Step (c).

Now, we are able to define the inner algorithm. Its iterates will be interior points of A except, perhaps, the last one. The algorithm "terminates" when a first-order stationary point x is found ($\bar{g}(x) = 0$) that is close to the boundary (distance smaller than $2\Delta_{\min}$), when a second-order stationary point x is found ($\bar{g}(x) = 0$ and $\nabla^2 \bar{f}(x)$ positive semidefinite) or when a point on the boundary is reached. If none of these possibilities takes place, the inner algorithm generates an infinite sequence.

Inner Algorithm

Given $\Delta_{\min} > 0$, $\Delta \geq \Delta_{\min}$, $0 < \lambda_{\min} < \lambda_{\max} < \infty$, $\lambda^k \in [\lambda_{\min}, \lambda_{\max}]$ and $x^k \in A^o$, a typical iteration of the inner algorithm is:

Step 1. *Compute distance to the boundary.*
Compute
$$\Delta_{\text{bound}} = \max\{\bar{\Delta} \geq 0 \mid \bar{B}(x^k, \bar{\Delta}) \subset A\}.$$

Step 2. *Close to the boundary, use SPG.*
If
$$\Delta_{\text{bound}} < 2\Delta_{\min} \tag{2}$$
and $\bar{g}(x^k) = 0$ terminate the execution of the Inner Algorithm declaring that x^k is a *"First-order stationary point close to the boundary"*. If (2) holds but $\bar{g}(x^k) \neq 0$, use an Inner SPG iteration to compute d^k and y^k and go to Step 6.

Step 3. *Far from the boundary solve the trust-region subproblem.*
(Note that this step is executed only if (2) does not hold.)

Step 3.1
Compute $B_k = \nabla^2 \bar{f}(x^k)$.

Step 3.2
Compute x^{trial}, a solution of the trust-region subproblem. If $\psi_k(x^{\text{trial}}) = 0$ terminate the execution of the Inner Algorithm declaring that x^k is a *"Second-order stationary point"*.
If $x^{\text{trial}} \notin A^o$, compute
$$t_{\max} = \max\{t \in [0,1] \mid [x^k, x^k + t\ (x^{\text{trial}} - x^k)] \subset A\}.$$

If $\bar{f}(x^k + t_{\max}(x^{\text{trial}} - x^k)) < \bar{f}(x^k)$ define $y^k = x^k + t_{\max}(x^{\text{trial}} - x^k)$ and go to Step 6. If $x^{\text{trial}} \notin A^o$ but $\bar{f}(x^k + t_{\max}(x^{\text{trial}} - x^k)) \geq \bar{f}(x^k)$, choose $\Delta \in [\Delta_{\min}, \Delta_{bound})$ and repeat Step 3.2. (Observe that, now, the solution x^{trial} of the new trust-region subproblem will necessarily be interior to A.)

Step 4. *Acceptance or rejection of the trust-region subproblem solution.*
Define
$$Pred = -\psi_k(x^{\text{trial}}), \quad Ared = \bar{f}(x^k) - \bar{f}(x^{\text{trial}}).$$
If
$$Ared \geq 0.1\ Pred \tag{3}$$
define $y^k = x^{\text{trial}}$. Otherwise, choose
$$\Delta \in [0.1\ \|x^{\text{trial}} - x^k\|, 0.9\ \|x^{\text{trial}} - x^k\|]$$
and go to Step 3.2.

Step 5. *Update the trust radius.*

Choose $\Delta \geq \Delta_{\min}$.

Step 6. *Possible additional improving.*
Choose $x^{k+1} \in A$ such that $\bar{f}(x^{k+1}) \leq \bar{f}(y^k)$. (Observe that $x^{k+1} = y^k$ is an admissible choice.) If x^{k+1} belongs to the boundary of A, terminate the execution of the Inner Algorithm declaring *"Iterate on the boundary"*.

Remark. The addition of Step 6, with the possible choice of $x^{k+1} \neq y^k$ has, essentially, theoretical purposes. We will realise the importance of such assumption in the proof of the convergence theorem of the next section. However, the freedom allowed by the choice of x^{k+1} can be also used for improving the iteration using extrapolation techniques. We will comment this feature in the section of Numerical Experiments.

Theorem 1. *In the execution of the Inner Algorithm, one, and only one of the following possibilities takes place:*

1. *The algorithm terminates in a finite number of iterations at a point x^k such that its distance to the boundary is less than $2\Delta_{\min}$ and $\bar{g}(x^k) = 0$;*

2. *The algorithm terminates in a finite number of iterations at a second-order stationary point x^k such that $\Delta_{\text{bound}} \geq 2\Delta_{\min}$;*

3. *The algorithm terminates after a finite number of iterations at a point x^{k+1} on the boundary of A and*
$$\bar{f}(x^{k+1}) < \bar{f}(x^k) < \ldots < \bar{f}(x^0).$$

4. *The algorithm generates a sequence of infinitely many iterates. In this case: (a) Every limit point x^* of the sequence so far generated satisfies $\bar{g}(x^*) = 0$. (b) If a limit point x^* is such that its distance to the boundary is greater than $2\Delta_{\min}$ then $\nabla^2 \bar{f}(x^*)$ is positive semidefinite.*

Proof. The first three possibilities are considered in the definition of the algorithm, corresponding to the three different termination criteria. Suppose that an infinite sequence is generated and that x^* is a limit point of that sequence. So, there exists an infinite set $K_1 \subset I\!N$ such that
$$\lim_{k \in K_1} x^k = x^*.$$
We consider two possibilities:

(a) For infinitely many iterations $k \in K_2 \subset K_1$, x^{k+1} is computed at Step 2.

(b) For infinitely many iterations $k \in K_3 \subset K_1$, x^{k+1} is computed at Step 3.

In the case (a) the arguments that lead to the proof of the convergence of the SPG method [4, 6] show that $\bar{g}(x^*) = 0$. In fact, if we assume that $\bar{g}(x^*) \neq 0$ and that t_k is bounded away from zero we get the contradiction $\bar{f}(x^k) \to -\infty$. On the other hand, if the infimum of $\{t_k\}$ is zero, we have an auxiliary sequence t'_k that tends to zero and such that the test (1) is rejected then $t = t'_k$. This implies, using classical arguments, that $\bar{g}(x^*) = 0$.

In the case (b), after some relabeling, the sequence can be interpreted as a sequence of trust-region iterates (with the initial trust radius Δ being not smaller than $\Delta_{\min} > 0$) where each iterate is modified with a further reduction of the objective function. So, classical arguments of the trust-region literature (see, for example, [18, 19]) imply that $\bar{g}(x^*) = 0$ and $\nabla^2 \bar{f}(x^*)$ positive semidefinite.

The arguments above imply that any limit point x^* satisfies $\bar{g}(x^*) = 0$. Moreover, if the distance of x^* to the boundary is greater than $2\Delta_{\min}$, then for $k \in K_1$ large enough, x^{k+1} is computed at Step 3. Therefore, the case (b) applies and, so, the Hessian is positive semidefinite. $\square$

Remark. Pure Euclidian trust-region methods for bound-constrained minimization are not admissible. Near the boundary, something different must be done in order to obtain convergence. The strategy chosen in the Inner Algorithm is one of the possible strategies that lead to suitable convergence properties. In order to understand why the trust-region strategy alone does not work, consider the problem

$$\text{Minimize } x_1 + x_2$$

subject to

$$-1 \leq x_1 \leq 100, \; 0 \leq x_2 \leq 100.$$

The solution of this problem is $(-1, 0)$. Suppose that $x^0 = (1, 1)$ and that x^{trial} is the solution of the Euclidian trust-region subproblem defined by the distance of x^k to the boundary. The test (3) is obviously satisfied by x^{trial} for all k so we can define $x^{k+1} = x^{\text{trial}}$. However, all the iterates are in the line $x_1 = x_2$, the sequence is infinite and converges to $(0, 0)$ which is not an interior point and $\bar{g}(0, 0) = (1, 1) \neq (0, 0)$.

3 Algorithm BETRA

In this section

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

and $f : I\!R^n \to I\!R$ has continuous third-order derivatives on an open set that contains Ω. We denote $g = \nabla f$. As in [3, 15] we divide the feasible set Ω into disjoint open faces. For all $I \subset \{1, 2, \ldots, n, n+1, n+2, \ldots 2n\}$ we define

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

The box Ω is the union of its open faces. The variables x_i such that neither i nor $n+i$ belong to I are called *free* in F_I. The Inner Algorithm defined in the previous section modifies only the free variables. Therefore, the closed convex set A will be identified with the different closures of the open faces. We define V_I the smallest affine subspace that contains F_I and S_I the parallel linear subspace to V_I. Accordingly, we define the projected gradient:

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

and, for all $x \in F_I$, we define the internal gradient:

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

Observe that $g_I(x) = 0$ when the face F_I is a vertex since, in that case, $S_I = \{0\}$.

As in [3], the box-constrained algorithm will combine inner iterations with SPG iterations. Global SPG iterations are similar to Inner SPG iterations defined in the previous section. However, the current iterate is not restricted to be interior to Ω.

Global Spectral Projected Gradient (SPG) iteration

Given $x^k \in \Omega$, $0 < \lambda_{\min} < \lambda_{\max} < \infty$, $\lambda^k \in [\lambda_{\min}, \lambda_{\max}]$, $\alpha \in (0, \frac{1}{2})$, and assuming that $g_P(x^k) \neq 0$, perform the following steps:

(a) Compute

$$d^k = P_\Omega(x^k - \lambda^k g(x^k)) - x^k.$$

($g_P(x^k) \neq 0$ implies that $d^k \neq 0$.)

(b) Set $t \leftarrow 1$.

(c) If

$$f(x^k + t\, d^k) \leq f(x^k) + \alpha\, t\, g(x^k)^T d^k \tag{4}$$

set $t_k = t$ and terminate the iteration defining $x^{k+1} = x^k + t_k d^k$. If (4) does not hold, choose $t_{new} \in [0.1\, t, 0.5\, t]$, set $t \leftarrow t_{new}$ and repeat Step (c).

Let us now define the box-Euclidian-trust-region algorithm (BETRA).

Algorithm BETRA

Given $\eta \in (0, 1)$, $0 < \lambda_{\min} < \lambda_{\max} < \infty$, and $x^k \in \Omega$, the iteration that defines x^{k+1} or terminates the execution of BETRA is as follows:

Step 1.

Let F_I be the open face to which x^k belongs. Compute $g_P(x^k)$ and $g_I(x^k)$. If F_I is a vertex (the number of elements of I is n) and $g_P(x^k) = 0$, terminate the execution of BETRA.

Step 2.

Compute the spectral steplength

$$\lambda^k = \begin{cases} \min\{\lambda_{\max}, \max\{\lambda_{\min}, \frac{\|x^k - x^{k-1}\|^2}{(x^k - x^{k-1})^T (g_k - g_{k-1})}\}\}, & \text{if } (x^k - x^{k-1})^T (g_k - g_{k-1}) > 0, \\ \lambda_{\max} & \text{otherwise.} \end{cases} \tag{5}$$

The arbitrary initial spectral steplength $\lambda^0 \in [\lambda_{\min}, \lambda_{\max}]$ is computed by the spectral formula (5) just replacing x^{k-1} by $(x^0 - t_{\text{small}}\, g_0)$, where t_{small} is a small number.

Step 3.

If

$$\|g_I(x^k)\| \geq \eta \|g_P(x^k)\|, \tag{6}$$

assume, without loss of generality, that the free variables at F_I are $x_1, \ldots, x_m$, and that the remaining variables are fixed in $\bar{x}_{m+1}, \ldots, \bar{x}_n$. ($\bar{x}_i \in \{\ell_i, u_i\}\ \forall i = m+1, \ldots, n$.) Define

$$\bar{f}(x_1, \ldots, x_m) = f(x_1, \ldots, x_m, \bar{x}_{m+1}, \ldots, \bar{x}_n)$$

for all $x_1, \ldots, x_m \in I\!R$ and

$$A = \{x \in I\!R^m \mid \ell_i \le x_i \le u_i\ \forall\ i = 1, \ldots, m\}.$$

Perform an Inner Algorithm iteration with λ^k defined by the spectral formula (5). If the Inner Algorithm iteration terminates at the current point x^k declaring *"First-order stationary point close to the boundary"* or *"Second-order stationary point"*, terminate the execution of BETRA with the same declarations. (In both cases, due to (6), $g_P(x^k) = 0$.) Else, the Inner Algorithm iteration computes x^{k+1} and, on return, the iteration of BETRA finishes.

Step 4.
If (6) does not hold (so, $g_P(x^k) \neq 0$), perform an iteration of the Global SPG algorithm with λ^k defined by the spectral formula (5) for computing x^{k+1} and finish the iteration of BETRA.

Remarks. If the current face is not a vertex, the algorithm BETRA does not necessarily terminates when, at the beginning of an iteration, $g_P(x^k) = 0$. In this case, the test (6) necessarily holds, so an inner iteration is tried. If x^k is not close to the boundary, a trust-region iteration will be tried with the aim of finding a second-order stationary point.

Theorem 2. *One of the following possibilities hold:*

1. *The execution of BETRA terminates at an iteration where the Inner Algorithm terminates with $g_P(x^k) = 0$. In this case, if $x^k \in F_I$ and the distance between x^k and the boundary of F_I is greater than $2\Delta_{\min}$, the matrix $\nabla^2 f(x^k)$ is positive semidefinite when restricted to F_I.*

2. *BETRA generates an infinite sequence of iterates and at least one limit point of this sequence is a first-order stationary point.*

In the second case, if all the limit points are nondegenerate, all the iterates belong, eventually, to the same open face F_I. In this case, if the distance of a limit point x^* to the boundary of F_I is greater than $2\Delta_{\min}$, then $\nabla^2 f(x^*)$ is positive semidefinite when restricted to F_I. Moreover, if this reduced matrix is positive definite, the sequence converges quadratically to x^*.

Proof. By the definition of BETRA we see that finite termination only occurs when x^k is a vertex and $g_P(x^k) = 0$ or when the Inner Algorithm terminates with the declarations *"First-order stationary point close to the boundary"* or *"Second-order stationary point"*. In both cases, we have that the derivatives corresponding to free variables are null, so $g_I(x^k) = 0$. Then, by (6), $g_P(x^k) = 0$. By Theorem 1, the reduced Hessian is positive semidefinite if the distance between x^k and the boundary of its face is greater than $2\Delta_{\min}$.

It remains to consider the case in which infinitely many iterates are generated. We consider two possibilities:

(a) There exists $K_1 \subset I\!\!N$ an infinite set of indices such that x^{k+1} is computed by a global SPG iteration for all $k \in K_1$.

(b) For all $k \in I\!\!N$ large enough, x^{k+1} is computed by the Inner Algorithm.

If (a) takes place, we prove, as in [3], that every limit point of $\{x^k\}_{k\in K_1}$ must be first-order stationary.

If (b) holds, there exists $k_0 \in I\!\!N$ such that x^{k+1} is computed by the Inner Algorithm for all $k \geq k_0$. Since the number of faces is finite and active constraints are abandoned only at global SPG iterations, there exists a face F_I such that $x^k \in F_I$ for all k large enough. Therefore, the thesis of Theorem 1 applies and, so, $\lim_{k\to\infty} g_I(x^k) = 0$. So, by (6), $\lim_{k\to\infty} g_P(x^k) = 0$. This implies that all the limit points satisfy $g_P(x^*) = 0$. Also by Theorem 1, if the distance to the boundary of a limit point is greater than $2\Delta_{\min}$, the reduced Hessian is positive semidefinite. In this case, if the Hessian is positive definite, quadratic convergence follows as in the classical theory of the Newtonian unconstrained trust-region method (see [23]). The fact that in the nondegenerate case all the iterates eventually belong to the same face follows as in Theorem 3.4 of [2]. $\square$

4 Numerical experiments

In the SPG iterations, the computation of t_{new} uses one-dimensional quadratic interpolation and it was safeguarded taking $t_{\text{new}} \leftarrow t/2$ when the minimum of the one-dimensional quadratic lies outside $[0.1, 0.5\, t]$.

The computation of the initial trust radius Δ depends on a parameter Δ_{initial} in the following way:

$$\Delta = \max\{\Delta_{\min}, \Delta_{\text{initial}} \max\{1, \|x^0\|\}\}.$$

At Step 4 of the Inner Algorithm, when (3) does not hold, we choose

$$\Delta = \frac{1}{4}\|x^{\text{trial}} - x^k\|.$$

At Step 5 of the Inner Algorithm, the trust radius is updated based on the relation between the reduction obtained in the objective function and the one predicted by its quadratic model in the following way (see [13]):

$$\bar{\Delta} = \begin{cases} \frac{1}{4}\|x^{\text{trial}} - x^k\|, & \text{if } \frac{Ared}{Pred} \leq \frac{1}{4}, \\ 2\Delta, & \text{if } \frac{Ared}{Pred} \geq \frac{1}{2} \text{ and } \mid \|x^{\text{trial}} - x^k\| - \Delta \mid \leq 10^{-5}, \\ \Delta, & \text{otherwise}, \end{cases}$$

and

$$\Delta = \max\{\Delta_{\min}, \bar{\Delta}\}.$$

To solve the trust-region subproblem at Step 3 of the Inner Algorithm, we implemented the trust-region algorithm of Moré and Sorensen proposed in [21].

Whenever we need to choose $\Delta \in [\Delta_{\min}, \Delta_{\text{bound}})$ at Step 3 of the Inner Algorithm, we define

$$\Delta = \max\left\{\Delta_{\min}, \Delta_{\min} + 0.9\left(\left(\frac{\Delta_{\text{bound}}}{1+\sigma}\right) - \Delta_{\min}\right)\right\}, \tag{7}$$

where σ is a parameter of the Moré-Sorensen trust-region algorithm. Using this algorithm, the solution x^{trial} may be outside the trust region bounded by Δ, but the distance from this solution to the trust region is, at most, σ. The choice (7) of Δ ensures that, repeating Step 3 of the Inner Algorithm, we will find a solution of the trust-region subproblem that belongs to A^o.

At Step 6 of the Inner Algorithm, where we can have an additional improvement, we used the extrapolation technique presented in [3]. We decide to extrapolate when

$$(d^k)^T \bar{g}(x^k + d^k) < 0.5(d^k)^T \bar{g}(x^k),$$

where d^k is the direction computed by SPG or $d^k = x^{\text{trial}} - x^k$, when we solve the trust-region subproblem. The extrapolation depends on a parameter N used to multiply the current step.

The computer version of BETRA has the following stopping criteria:

1. *First-order stationary vertex* (Step 1 of BETRA):

   The current face is a vertex and $\|g_P(x^k)\|_\infty \leq \varepsilon$;

2. *First-order stationary point close to the boundary* (Step 2 of the Inner Algorithm):

   The inequality (2) holds and $\|\bar{g}(x^k)\|_\infty \leq \varepsilon$;

3. *Second-order stationary point* (Step 3 of the Inner Algorithm):

   The point x^{trial} is the solution of the trust-region subproblem, $|\psi_k(x^{\text{trial}})| < \varepsilon$ and $\|\bar{g}(x^k)\|_\infty < \varepsilon$;

4. *First-order stationary point* (Step 4 of BETRA):

   The test (6) does not hold (the current face should be abandoned) and $\|g_P(x^k)\|_\infty < \varepsilon$.

In the experiments we used $\sigma_1 = \sigma$, $\sigma_2 = 0$, $\lambda_0 = 0$, and $\epsilon = 10^{-5}$ for the algorithm of Moré and Sorensen (See [21] for details on these parameters); $\alpha = 10^{-4}$, $\lambda_{\min} = 10^{-10}$, and $\lambda_{\max} = 10^{10}$ for SPG; and $\varepsilon = 10^{-5}$.

To set the remaining parameters: η, Δ_{initial}, $\Delta_{\min}$, σ, and N; we tested all the combinations of $\eta \in \{0.1, 0.9\}$, $\Delta_{\text{initial}} \in \{0.1, 1, 10, 100\}$, $\Delta_{\min} \in \{10^{-5}, 10^{-4}, 10^{-3}\}$, $\sigma \in \{0.1, 0.2\}$, and $N \in \{2, 4\}$, in a reduced set of problems. The combination that provided the best results was $\eta = 0.1$, $\Delta_{\text{initial}} = 100$, $\Delta_{\min} = 10^{-4}$, $\sigma = 0.2$, and $N = 4$. Extrapolations are done when Step 6 is reached coming from Steps 3 and 5.

In order to assess the performance of BETRA, we compared it against LANCELOT [10]. In LANCELOT we used true second derivatives, and full-matrix preconditioners. Apart of this, we used the default options. To perform the numerical experiments, we considered all the (140) bound-constrained problems of the CUTE collection [7] with up to 120 variables.

All experiments were done in a Pentium I, with 133 MHz, 64 Mb of RAM memory, running Linux. The codes were written in FORTRAN 77 and compiled with g77 - GNU project Fortran Compiler (v0.5.24). We used the option -O3 to optimize the code. As the execution times are very small, we run each pair method/problem "several times" and report the average CPU time to minimize measurement errors.

Tables 1, 2 and 3 show the performance of both methods in the whole set of problems. In the tables, the columns mean: *Problem (n)* is the problem name and the number of variables between brackets; *iTR* is the number of trust-regions iterations; *nch* is the number of Cholesky decompositions; *iSPGg* is the number of global SPG iterations; *iSPGi* is the number of inner SPG iterations; *iTotal* is the total number of iterations; (in these columns, the numbers in brackets are the numbers of functional evaluations); *nfe* is the number of functional evaluations; *nge* is the number of gradient evaluations; *ncg* is the number of Conjugate Gradient iterations (used by LANCELOT); *Time* is the CPU time in seconds; $f(x^*)$ is the functional value at the best solution found. The number of Hessian evaluations performed by LANCELOT is *nge* - 1. The double lines in the tables divide the problems according to their objective function type. The objective functions of the problems in the first group are quadratic, those in the second group are sum of squares, and those in the third group are nonlinear general functions.

Table 1: BETRA and Lancelot performance in problems with less than 10 variables.

	BETRA										LANCELOT				
Problem (n)	iTR		nch	iSPGg		iSPGi	iTotal		Time	$f(x^*)$	nfe	nge	ncg	Time	$f(x^*)$
BQP1VAR (1)	2	(2)	1	0		0	2	(2)	0.000	0.0000E+00	3	3	0	0.004	0.0000E+00
HS3 (2)	3	(3)	4	0		0	3	(3)	0.001	7.8886E−36	8	8	0	0.009	9.5614E−37
HS3MOD (2)	3	(3)	5	0		0	3	(3)	0.001	7.8886E−31	4	4	3	0.008	4.1591E−32
OSLBQP (8)	2	(2)	2	0		0	2	(2)	0.001	6.2500E+00	3	3	0	0.008	6.2500E+00
SIM2BQP (2)	1	(1)	0	1	(1)	0	2	(2)	0.000	0.0000E+00	3	3	0	0.005	0.0000E+00
SIMBQP (2)	4	(4)	4	1	(1)	0	5	(5)	0.001	2.4074E−34	2	2	0	0.005	1.1132E−30
HATFLDA (4)	13	(13)	16	1	(8)	0	14	(21)	0.005	1.8367E−14	28	28	22	0.037	2.7495E−14
HATFLDB (4)	10	(10)	13	1	(8)	0	11	(18)	0.004	5.5728E−03	25	25	20	0.033	5.5728E−03
HS1 (2)	25	(29)	39	0		0	25	(29)	0.005	4.3945E−15	33	28	23	0.033	1.2242E−12
HS2 (2)	6	(6)	6	0		0	6	(6)	0.001	4.9412E+00	7	7	2	0.010	4.9412E+00
HS25 (3)	1	(1)	1	0		0	1	(1)	0.013	3.2835E+01	1	1	0	0.013	3.2835E+01
PALMER1 (4)	28	(34)	67	0		0	28	(34)	0.069	1.1755E+04	26	22	14	0.080	1.1755E+04
PALMER1A (6)	80	(87)	136	0		0	80	(87)	0.255	8.9883E−02	78	68	61	0.308	8.9883E−02
PALMER1B (4)	71	(79)	116	0		0	71	(79)	0.165	3.4473E+00	44	38	25	0.119	3.4473E+00
PALMER1E (8)	80	(86)	132	1	(16)	0	81	(102)	0.441	8.3523E−04	5	5	5	0.034	3.1103E+00
PALMER2 (4)	16	(21)	34	0		0	16	(21)	0.033	3.6511E+03	28	23	12	0.068	3.6511E+03
PALMER2A (6)	47	(50)	84	0		0	47	(50)	0.109	1.7110E−02	100	86	94	0.316	1.7110E−02
PALMER2B (4)	51	(58)	95	0		0	51	(58)	0.081	6.2327E−01	25	22	15	0.060	6.2327E−01
PALMER2E (8)	76	(81)	116	0		0	76	(81)	0.282	2.0650E−04	315	261	320	1.357	2.0650E−04
PALMER3A (6)	89	(95)	143	0		0	89	(95)	0.209	2.0431E−02	89	76	86	0.286	2.0431E−02
PALMER3B (4)	41	(47)	67	0		0	41	(47)	0.068	4.2276E+00	28	23	22	0.067	4.2276E+00
PALMER3E (8)	48	(53)	92	0		0	48	(53)	0.182	5.0741E−05	57	47	57	0.251	5.0741E−05
PALMER4 (4)	16	(18)	33	2	(6)	0	18	(24)	0.030	2.2854E+03	47	38	22	0.117	2.2854E+03
PALMER4A (6)	38	(42)	82	0		0	38	(42)	0.085	4.0606E−02	67	56	65	0.211	4.0606E−02
PALMER4B (4)	50	(55)	84	0		0	50	(55)	0.086	6.8351E+00	26	22	18	0.063	6.8351E+00
PALMER4E (8)	48	(50)	80	0		0	48	(50)	0.171	1.4800E−04	40	32	40	0.176	1.4800E−04
PALMER5A (8)	10000	(10029)	10474	0		0	10000	(10029)	20.280	4.0560E−02	10001	8402	9975	27.250	3.1022E−02
PALMER5B (9)	627	(646)	886	0		0	627	(646)	1.326	9.7524E−03	123	101	114	0.377	9.7524E−03
PALMER5D (8)	3	(3)	5	0		0	3	(3)	0.002	8.7339E+01	2	2	1	0.010	8.7339E+01
PALMER5E (8)	1428	(1437)	1511	0		0	1428	(1437)	3.015	2.2937E−02	8763	7212	8762	24.970	2.0716E−02
PALMER6A (6)	117	(124)	188	0		0	117	(124)	0.171	5.5949E−02	141	120	134	0.317	5.5949E−02
PALMER6E (8)	80	(82)	111	0		0	80	(82)	0.185	2.2395E−04	61	50	62	0.189	2.2395E−04
PALMER7A (6)	10000	(10008)	10116	0		0	10000	(10008)	15.220	1.0345E+01	4007	3569	4008	8.910	1.0335E+01
PALMER7E (8)	6	(6)	17	0		0	6	(6)	0.013	1.0154E+01	10001	8065	9930	29.120	6.5607E+00
PALMER8A (6)	36	(38)	68	0		0	36	(38)	0.049	7.4010E−02	52	45	55	0.119	7.4010E−02
PALMER8E (8)	26	(27)	52	0		0	26	(27)	0.058	6.3393E−03	41	34	41	0.124	6.3393E−03
PSPDOC (4)	6	(8)	11	0		0	6	(8)	0.002	2.4142E+00	10	10	9	0.015	2.4142E+00
WEEDS (3)	5	(5)	10	2	(18)	0	7	(23)	0.007	9.2054E+03	25	22	27	0.050	2.5873E+00
YFIT (3)	50	(60)	93	0		0	50	(60)	0.086	6.6697E−13	51	42	50	0.112	6.6698E−13
ALLINIT (4)	10	(11)	13	1	(1)	0	11	(12)	0.007	1.6706E+01	12	10	6	0.017	1.6706E+01
CAMEL6 (2)	30	(37)	49	0		0	30	(37)	0.010	-1.0316E+00	19	11	12	0.020	-1.0316E+00
HART6 (6)	13	(14)	22	1	(1)	0	14	(15)	0.013	-3.3229E+00	9	7	7	0.019	-3.3229E+00
HIMMELP1 (2)	6	(6)	8	1	(3)	0	7	(9)	0.002	-6.2054E+01	16	15	5	0.014	-6.2054E+01
HS38 (4)	49	(55)	91	0		0	49	(55)	0.023	1.9971E−20	54	46	56	0.069	5.2378E−20
HS4 (2)	2	(2)	2	0		0	2	(2)	0.000	2.6667E+00	2	2	0	0.004	2.6667E+00
HS45 (5)	2	(2)	13	0		0	2	(2)	0.001	1.0000E+00	9	9	0	0.009	1.0000E+00
HS5 (2)	8	(9)	16	1	(1)	0	9	(10)	0.002	-1.9132E+00	6	6	3	0.009	-1.9132E+00
LOGROS (2)	54	(60)	127	2	(13)	0	56	(73)	0.018	0.0000E+00	56	45	35	0.054	0.0000E+00
MAXLIKA (8)	35	(36)	61	4	(20)	0	39	(56)	4.017	1.1363E+03	9	8	19	0.659	1.1493E+03
MDHOLE (2)	37	(39)	65	0		0	37	(39)	0.010	4.8148E−33	61	51	52	0.058	3.0080E−37
S368 (8)	6	(6)	8	0		0	6	(6)	0.026	-9.3750E−01	6	6	3	0.029	-9.3750E−01

Problem (n)	BETRA											LANCELOT				
	iTR		nch	iSPGg		iSPGi		iTotal		Time	$f(x^*)$	nfe	nge	ncg	Time	$f(x^*)$
BIGGSB1 (25)	15	(15)	15	12	(27)	0		27	(42)	0.026	1.5000E−02	15	15	14	0.041	1.5000E−02
BQPGABIM (50)	1	(1)		1	(2)	43	(57)	45	(60)	0.053	-3.7903E−05	3	3	4	0.042	-3.7903E−05
BQPGASIM (50)	1	(1)		1	(1)	42	(54)	44	(56)	0.054	-5.5198E−05	3	3	3	0.040	-5.5198E−05
CHENHARK (10)	5	(5)	5	2	(3)	0		7	(8)	0.004	-2.0000E+00	2	2	5	0.011	-2.0000E+00
CVXBQP1 (10)	2	(2)	3	0		1	(1)	3	(3)	0.001	2.4750E+00	2	2	0	0.009	2.4750E+00
HARKERP2 (10)	9	(9)	9	0		0		9	(9)	0.011	-5.0000E−01	3	3	1	0.011	-5.0000E−01
JNLBRNG1 (16)	2	(2)	2	0		0		2	(2)	0.002	-2.2474E−01	2	2	2	0.012	-2.2474E−01
JNLBRNG2 (16)	2	(2)	2	0		0		2	(2)	0.002	-4.7640E+00	2	2	0	0.011	-4.7640E+00
JNLBRNGA (16)	2	(2)	2	1	(1)	0		3	(3)	0.002	-5.0967E−01	5	5	0	0.013	-5.0967E−01
JNLBRNGB (16)	2	(2)	2	1	(1)	0		3	(3)	0.002	-1.8551E+01	5	5	1	0.013	-1.8551E+01
NCVXBQP1 (10)	2	(2)	1	0		0		2	(2)	0.001	-2.2050E+04	3	3	0	0.010	-2.2050E+04
NCVXBQP2 (10)	2	(2)	1	0		0		2	(2)	0.001	-1.4382E+04	3	3	0	0.010	-1.4382E+04
NCVXBQP3 (10)	2	(2)	1	1	(1)	0		3	(3)	0.001	-1.1958E+04	3	3	0	0.010	-1.1958E+04
NOBNDTOR (16)	2	(2)	2	0		0		2	(2)	0.001	-5.4321E−01	3	3	0	0.011	-5.4321E−01
OBSTCLAE (16)	2	(2)	1	0		0		2	(2)	0.001	7.5366E−01	3	3	0	0.010	7.5366E−01
OBSTCLAL (16)	1	(1)		0		0		1	(1)	0.000	7.5366E−01	1	1	0	0.008	7.5366E−01
OBSTCLBL (16)	1	(1)		1	(1)	0		2	(2)	0.000	-8.1108E−03	2	2	0	0.009	-8.1108E−03
OBSTCLBM (16)	2	(2)	1	0		0		2	(2)	0.001	-8.1108E−03	2	2	0	0.010	-8.1108E−03
OBSTCLBU (16)	1	(1)		0		0		1	(1)	0.000	-8.1108E−03	1	1	0	0.009	-8.1108E−03
PENTDI (50)	2	(2)	2	1	(1)	0		3	(3)	0.007	-7.5000E−01	2	2	0	0.023	-7.5000E−01
QUDLIN (12)	1	(1)		1	(1)	0		2	(2)	0.000	-7.2000E+03	2	2	0	0.011	-7.2000E+03
TORSION1 (16)	1	(1)		0		0		1	(1)	0.000	-5.1852E−01	1	1	0	0.009	-5.1852E−01
TORSION2 (16)	2	(2)	1	0		0		2	(2)	0.001	-5.1852E−01	3	3	0	0.010	-5.1852E−01
TORSION3 (16)	1	(1)		0		0		1	(1)	0.000	-1.2593E+00	1	1	0	0.008	-1.2593E+00
TORSION4 (16)	2	(2)	1	0		0		2	(2)	0.001	-1.2593E+00	3	3	0	0.010	-1.2593E+00
TORSION5 (16)	1	(1)		0		0		1	(1)	0.000	-2.7407E+00	1	1	0	0.008	-2.7407E+00
TORSION6 (16)	2	(2)	1	0		0		2	(2)	0.001	-2.7407E+00	2	2	0	0.009	-2.7407E+00
TORSIONA (16)	2	(2)	2	1	(1)	0		3	(3)	0.003	-3.0864E−01	2	2	0	0.011	-3.0864E−01
TORSIONB (16)	2	(2)	2	0		0		2	(2)	0.003	-3.0864E−01	3	3	0	0.012	-3.0864E−01
TORSIONC (16)	1	(1)		0		0		1	(1)	0.000	-1.0370E+00	1	1	0	0.008	-1.0370E+00
TORSIOND (16)	2	(2)	1	0		0		2	(2)	0.002	-1.0370E+00	3	3	0	0.011	-1.0370E+00
TORSIONE (16)	1	(1)		0		0		1	(1)	0.000	-2.5185E+00	1	1	0	0.009	-2.5185E+00
TORSIONF (16)	2	(2)	1	0		0		2	(2)	0.002	-2.5185E+00	2	2	0	0.010	-2.5185E+00
CHEBYQAD (50)	22	(27)	103	0		0		22	(27)	11.300	5.3863E−03	106	87	108	43.310	5.3863E−03
DECONVB (61)	23	(24)	52	6	(21)	7771	(11961)	7800	(12006)	29.894	5.3548E−08	20	16	128	0.852	5.6636E−03
HATFLDC (25)	5	(5)	5	0		0		5	(5)	0.013	8.2639E−14	5	5	3	0.026	7.7700E−19
HS110 (10)	6	(7)	8	0		0		6	(7)	0.009	-4.5778E+01	5	5	0	0.016	-4.5778E+01
LINVERSE (19)	6	(6)	9	0		0		6	(6)	0.017	7.0000E+00	10	8	16	0.055	7.0000E+00
NONSCOMP (50)	8	(8)	7	0		15	(31)	23	(39)	0.078	1.1916E−11	9	9	8	0.065	1.1212E−16
QR3DLS (40)	20	(23)	43	0		0		20	(23)	0.324	5.5383E−16	29	24	28	0.477	1.2807E−13
EXPLIN (12)	9	(9)	14	1	(2)	0		10	(11)	0.006	-6.8500E+03	13	11	13	0.024	-6.8500E+03
EXPLIN2 (12)	9	(9)	16	2	(4)	0		11	(13)	0.007	-7.0925E+03	14	13	16	0.027	-7.0925E+03
EXPQUAD (12)	10	(10)	45	1	(1)	0		11	(11)	0.019	-4.2011E+03	13	11	20	0.033	-4.2011E+03
HADAMALS (16)	17	(17)	22	0		0		17	(17)	0.045	0.0000E+00	8	8	37	0.046	0.0000E+00
MCCORMCK (10)	7	(7)	11	0		0		7	(7)	0.008	-9.5980E+00	5	5	4	0.016	-9.5980E+00
PROBPENL (10)	19	(27)	62	0		2	(7)	21	(34)	0.030	-3.1787E+05	5	3	13	0.020	1.5235E−05
QRTQUAD (12)	20	(22)	30	2	(13)	0		22	(35)	0.020	-3.6077E+03	71	56	64	0.117	-3.6077E+03
SINEALI (20)	10000	(10221)	11627	0		0		10000	(10221)	23.720	-1.9010E+03	10001	8887	9994	26.070	-1.9010E+03

Table 2: BETRA and Lancelot performance in problems with more than 10 and less than 100 variables.

Table 3: BETRA and Lancelot performance in problems with more than 100 variables.

	BETRA							LANCELOT				
Problem (n)	iTR	nch	iSPGg	iSPGi	iTotal	Time	$f(x^*)$	nfe	nge	ncg	Time	$f(x^*)$
BIGGSB1 (100)	52 (52)	52	50 (231)	1 (1)	103 (284)	0.950	1.5000E−02	52	52	50	0.323	1.5000E−02
CHENHARK (100)	3 (3)	6	0	0	3 (3)	0.145	-2.0000E+00	25	25	61	0.279	-2.0000E+00
CVXBQP1 (100)	2 (2)	3	0	1 (1)	3 (3)	0.104	2.2725E+02	2	2	0	0.055	2.2725E+02
HARKERP2 (100)	7 (7)	7	0	0	7 (7)	12.570	-5.0000E−01	2	2	2	0.223	-5.0000E−01
JNLBRNG1 (100)	3 (3)	3	1 (2)	0	4 (5)	0.050	-1.7896E−01	2	2	1	0.056	-1.7896E−01
JNLBRNG2 (100)	3 (3)	3	1 (1)	0	4 (4)	0.046	-3.9528E+00	3	3	2	0.066	-3.9528E+00
JNLBRNGA (100)	3 (3)	3	2 (3)	0	5 (6)	0.044	-3.6116E−01	3	3	2	0.061	-3.6116E−01
JNLBRNGB (100)	2 (2)	2	1 (1)	0	3 (3)	0.026	-7.2552E+00	4	4	3	0.069	-7.2552E+00
NCVXBQP1 (100)	2 (2)	1	0	0	2 (2)	0.045	-1.9956E+06	2	2	0	0.057	-1.9956E+06
NCVXBQP2 (100)	4 (4)	6	1 (2)	0	5 (6)	0.058	-1.3330E+06	3	3	4	0.061	-1.3330E+06
NCVXBQP3 (100)	5 (5)	6	2 (2)	0	7 (7)	0.066	-6.7085E+05	4	4	4	0.064	-6.7085E+05
NOBNDTOR (100)	4 (4)	4	2 (4)	0	6 (8)	0.060	-5.5211E−01	3	3	2	0.067	-5.5211E−01
OBSTCLAE (100)	5 (5)	5	3 (4)	0	8 (9)	0.079	1.3979E+00	3	3	29	0.138	1.3979E+00
OBSTCLAL (100)	4 (4)	4	3 (5)	0	7 (9)	0.060	1.3979E+00	4	4	3	0.071	1.3979E+00
OBSTCLBL (100)	5 (5)	5	2 (2)	0	7 (7)	0.056	2.8750E+00	3	3	6	0.067	2.8750E+00
OBSTCLBM (100)	3 (3)	3	0	0	3 (3)	0.043	2.8750E+00	2	2	3	0.060	2.8750E+00
OBSTCLBU (100)	3 (3)	3	1 (1)	0	4 (4)	0.036	2.8750E+00	2	2	1	0.053	2.8750E+00
TORSION1 (100)	3 (3)	3	2 (2)	0	5 (5)	0.040	-4.9234E−01	3	3	2	0.059	-4.9234E−01
TORSION2 (100)	4 (4)	4	2 (5)	0	6 (9)	0.061	-4.9234E−01	4	4	2	0.073	-4.9234E−01
TORSION3 (100)	2 (2)	2	1 (1)	0	3 (3)	0.022	-1.2705E+00	2	2	1	0.051	-1.2705E+00
TORSION4 (100)	3 (3)	3	1 (4)	0	4 (7)	0.046	-1.2705E+00	4	4	2	0.066	-1.2705E+00
TORSION5 (100)	1 (1)	0	0	0	1 (1)	0.002	-2.8971E+00	1	1	0	0.044	-2.8971E+00
TORSION6 (100)	2 (2)	1	0	0	2 (2)	0.025	-2.8971E+00	3	3	0	0.056	-2.8971E+00
TORSIONA (100)	3 (3)	3	2 (2)	0	5 (5)	0.050	-4.0570E−01	3	3	2	0.066	-4.0570E−01
TORSIONB (100)	5 (5)	5	0	0	5 (5)	0.095	-4.0570E−01	5	5	6	0.114	-4.0570E−01
TORSIONC (100)	2 (2)	2	1 (1)	0	3 (3)	0.027	-1.1766E+00	2	2	1	0.054	-1.1766E+00
TORSIOND (100)	3 (3)	3	1 (4)	0	4 (7)	0.054	-1.1766E+00	4	4	3	0.076	-1.1766E+00
TORSIONE (100)	1 (1)	0	0	0	1 (1)	0.002	-2.7984E+00	1	1	0	0.045	-2.7984E+00
TORSIONF (100)	2 (2)	1	0	0	2 (2)	0.029	-2.7984E+00	3	3	0	0.059	-2.7984E+00
HS110 (100)	2 (2)	2	0	0	2 (2)	0.109	-9.9800E+19	2	2	0	0.054	-9.9800E+19
NONSCOMP (100)	8 (8)	7	0	17 (39)	25 (47)	0.337	8.0101E−12	9	9	8	0.130	2.6015E−16
BDEXP (100)	12 (13)	16	0	0	12 (13)	0.696	1.3497E−05	11	11	10	0.192	3.9646E−05
EXPLIN (120)	25 (26)	31	3 (3)	0	28 (29)	0.107	-7.2376E+05	14	14	61	0.113	-7.2324E+05
EXPLIN2 (120)	13 (13)	61	4 (4)	0	17 (17)	0.056	-7.2446E+05	12	12	30	0.093	-7.2446E+05
EXPQUAD (120)	17 (18)	32	2 (4)	0	19 (22)	1.642	-3.6260E+06	18	15	49	0.354	-3.6260E+06
HADAMALS (100)	26 (26)	49	1 (3)	0	27 (29)	2.904	2.5316E+01	14	14	370	2.610	2.5316E+01
MCCORMCK (100)	7 (8)	12	0	0	7 (8)	0.472	-9.1788E+01	7	6	5	0.117	-9.1788E+01
PROBPENL (100)	4 (4)	13	0	0	4 (4)	0.673	-4.9571E−06	91	52	268	4.923	-2.8726E+05
QRTQUAD (120)	42 (44)	66	2 (12)	0	44 (56)	3.783	-3.6246E+06	205	168	195	2.469	-3.6242E+06
S368 (100)	7 (7)	10	0	0	7 (7)	6.630	-1.4869E+02	9	7	9	7.007	-1.3369E+02
SINEALI (100)	10 (11)	24	0	0	10 (11)	0.890	-9.9010E+03	13	9	6	0.138	-9.9010E+03

On average, 12.48% of the functional evaluations were done by the global SPG, 3.87% were done by the inner SPG and 83.65% were done by the trust-region Inner Algorithm. Out of the 140 tested problems, 33 problems (23.57%) satisfied criterion 1, 5 problems (3.57%) satisfied criterion 2, 98 (70.71%) satisfied criterion 3, and none satisfied criterion 4. The 3 remaining problems stopped because BETRA reached the maximum allowed number of iterations (10000). These problems were: PALMER5A that stopped with $\|g_P(x^k)\|_\infty =$ 1.8088E+00; PALMER7A, that stopped with $\|g_P(x^k)\|_\infty =$ 4.0094E−01; and SINEALI (20) that stopped with $\|g_P(x^k)\|_\infty =$ 2.6944E−03.

LANCELOT has only one successful stopping criterion: sup-norm of the projected gradient smaller than 10^{-5}. In 137 out of the 140 tested problems, this criteria was satisfied. In the 3 remaining problems, LANCELOT stopped because it reached the maximum allowed number of function evaluations (10000). These problems were: PALMER5A that stopped with $\|g_P(x^k)\|_\infty =$ 1.5959E−01; PALMER7E that stopped with $\|g_P(x^k)\|_\infty =$ 5.9967E+00; and SINEALI (20) that stopped with $\|g_P(x^k)\|_\infty =$ 7.9770E−04.

To have a better overview of the comparison between BETRA and LANCELOT, we used the CPU time as a performance measurement and constructed a performance profile graphic (see Figure 1). To construct this graphic, we took into account not only the CPU time spent by each method, but also the solution that each method provided. When both BETRA and LANCELOT provide a solution with the same functional value, we compare the CPU times. When they provide different solutions, we assume that the method that came up with the worst solution took "infinite" time. We consider that two functional values f_1 and f_2 are equivalent if

$$| f_1 - f_2 | \leq \max(10^{-10}, 10^{-6} \min(| f_1 |, | f_2 |)).$$

Figure 2 shows the performance profile curve just considering the problems in which both methods stopped by a successful criteria and found solutions with the same functional value (127 over the total of 140 problems).

In Figure 1, it can be seen that BETRA is faster than LANCELOT in 82.14% of the problems whereas LANCELOT is faster than BETRA in 19.28%. It can also be seen that BETRA successfully solved 96.43% of the problems whereas LANCELOT solved 95.71% of the problems. ("to be faster" and "to solve" a problem means what we explained above.) For the problems considered in Figure 2, BETRA is faster than LANCELOT in 85.04% of the cases and LANCELOT is faster than BETRA in 16.53% of the cases.

It is important to notice that, whenever the value of η is increased, the number of global SPG iterations tends to increase; and, whenever the value of $\Delta_{\min}$ is increased, the number of inner SPG iterations tends to increase too. Nevertheless, when the value of these two parameters are increased, BETRA stopped reaching a solution that satisfies the second-order necessary condition in just a few less problems. As an example, we tested the 140 problems changing the values of η and $\Delta_{\min}$. For $\eta = 0.01$ and $\Delta_{\min} = 10^{-5}$, the number of problems in which BETRA stopped satisfying criterion 1, 2, 3, 4 or did not satisfy any of them were, 33, 5, 97, 0, and 5, respectively. For $\eta = 0.99$ and $\Delta_{\min} = 10^{-3}$, the numbers were 33, 5, 96, 2, and 4, respectively. So, although these parameters are not so easy to choose, the robustness of BETRA does not strongly depend on them.

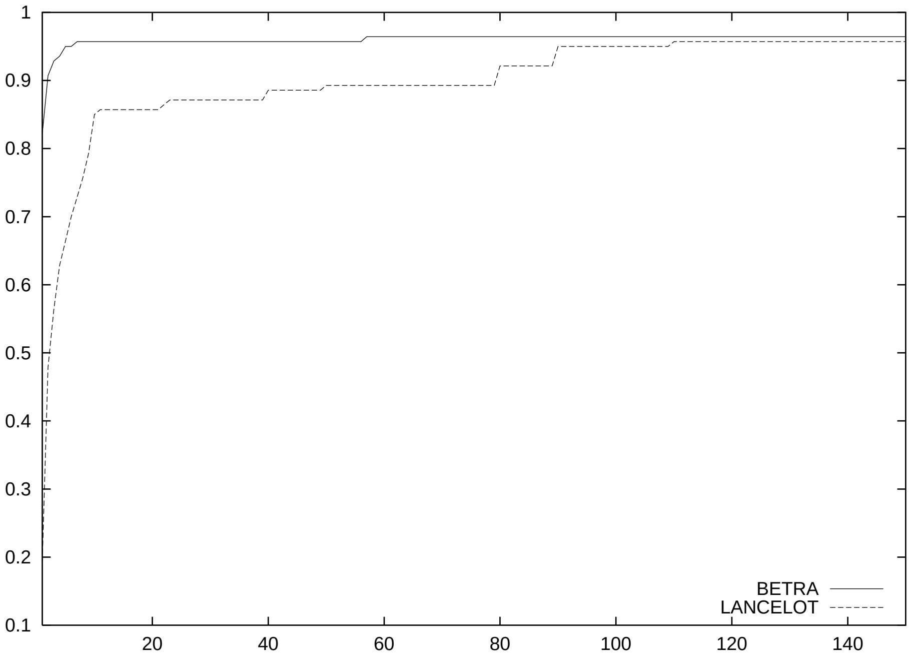

Figure 1: Performance profile curve comparing BETRA vs. Lancelot in the whole set of problems.

5 Conclusions

We presented a practical active-set method for solving bound-constrained smooth minimization problems, where the inner algorithm is the classical Euclidian trust-region method and the iterations for leaving faces are of spectral projected gradient type. Spectral projected gradient steps are also used when some inactive constraints at the current point are almost active. We prove convergence to first-order stationary points and, under suitable assumptions, to second-order stationary points. In the last case, the order of convergence results of second-order trust-region methods is maintained.

We compared the method against a well established algorithm that uses full Hessians and full preconditioners (so, Cholesky factorizations) for solving box-constrained problems. The results were encouraging. Probably, the new method is one of the most adequate ways of using trust-region strategies for bound-constrained problems, at least when the problems are not very large. For very large problems, the trust-region solution of the subproblem must be modified, for example, as in [22].

Euclidian and ellipsoidal trust regions have the advantage over box trust regions that *global* minimizers of the corresponding subproblems can be obtained using stable and efficient algorithms. On the other hand, balls and ellipsoids do not fit the shape of a box as well as trust-region boxes do. Therefore, the relative usefulness of both approaches in practical computations is a matter of controversy and, very likely, the correct decision is highly problem-dependent.

If one decides to use ellipsoidal trust regions the question about the shape of the ellipsoids

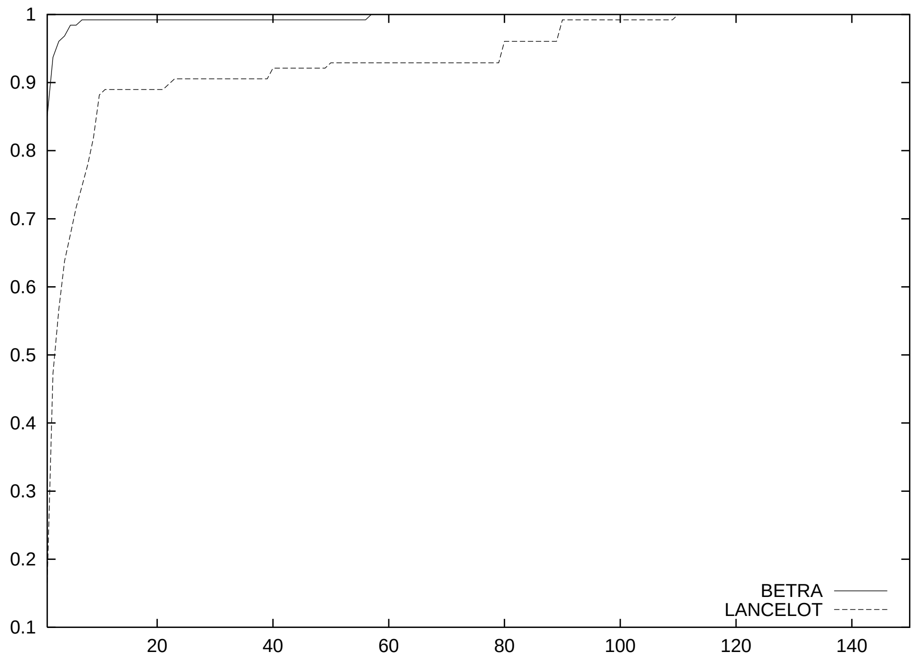

Figure 2: Performance profile curve comparing BETRA vs. Lancelot considering the problems in which both methods stopped satisfying a successful criteria and found the same functional value.

arises. Basically, there are two possibilities: either the shape of the ellipsoids depends on the constraint set or it depends only on the size of the variables. If the problem is badly scaled in the sense that independent variables are of very different magnitudes, it is probably better to pre-process the function so that the bad-scaling effect disappears as much as possible. Therefore, the choice is between constraint-shape ellipsoids and Euclidian balls.

In this paper, as in [11], we make our option for Euclidian balls-subproblems, but we also observed that there are no reasons for maintaining interiority of the iterates, as far as efficient leaving-face and extrapolation procedures for gaining and abandoning constraints exist. It seems that the numerical results confirm that this is a valid approach for general box-constrained minimization.

The extensions of the strategy presented in this paper are in two directions: first, we aim to consider large problems using large-scale trust-region solvers [22]. Second, we plan to extend the strategy to general polytopes.

Acknowledgement

The authors are indebted to an anonymous referee for helpful and encouraging comments.

References

[1] D. P. Bertsekas (1999) *Nonlinear Programming*, Athena Scientific, Belmont, MA.

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

[3] E. G. Birgin and J. M. Martínez (2002) Large-scale active-set box-constrained optimization method with spectral projected gradients, *Computational Optimization and Applications* **23**, 101–125.

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

[5] E. G. Birgin, J. M. Martínez and M. Raydan (2001) Algorithm 813: SPG - Software for convex-constrained optimization, *ACM Transactions on Mathematical Software* **27**, 340–349.

[6] E. G. Birgin, J. M. Martínez and M. Raydan (2003) Inexact spectral projected gradient methods on convex sets, *IMA Journal on Numerical Analysis* 23, 539–559.

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

[8] O. Burdakov, J. M. Martínez and E. A. Pilotta (2002) A limited memory multipoint secant method for bound constrained optimization, *Annals of Operations Research* **117**, 51–70.

[9] T. F. Coleman and Y. Li (1996) An interior trust-region approach for nonlinear minimization subject to bounds, *SIAM Journal on Optimization* **6**, 418–445.

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

[11] J. E. Dennis and L. N. Vicente (1996) Trust-region interior-point algorithms for minimization with simple bounds, in *Applied Mathematics and Parallel Computing*, edited by K. Ritter, H. Fisher, B. Riedmuller and S. Schaffer, Physica, Heidelberg, 97–107.

[12] E. D. Dolan and J. J. Moré (2002) Benchmarking optimization software with performance profiles, *Mathematical Programming* **91**, 201–213.

[13] R. Fletcher (1987) *Practical Methods of Optimization*, John Wiley & Sons, Chichester, New York, Brisbane, Toronto, Singapore.

[14] R. Fletcher (2001) On the Barzilai-Borwein method, *Department of Mathematics, University of Dundee* NA/207, Dundee, Scotland.

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

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

[17] D. M. Gay (1981) Computing optimal locally constrained steps, *SIAM Journal on Scientific and Statistical Computing* **2**, 186–197.

[18] J. M. Martínez and S. A. Santos (1995) A trust region strategy for minimization on arbitrary domains, *Mathematical Programming* **68**, 267–302.

[19] J. M. Martínez and S. A. Santos (1997) Convergence results on an algorithm for norm constrained regularization and related problems, *RAIRO Operations Research* **31**, 269–294.

[20] J. J. Moré (1983) Recent developments in algorithms and software for trust region methods, In: A. Bachem, M. Grötschel and B. Korte, eds., *Mathematical Programming Bonn 1982 - The State of Art*, Springer-Verlag.

[21] J. J. Moré and D. C. Sorensen (1983) Computing a trust region step, *SIAM Journal on Scientific and Statistical Computing* **4**, 553–572.

[22] M. Rojas, S. A. Santos and D. C. Sorensen (2000) A new matrix-free algorithm for the large-scale trust-region subproblem, *SIAM Journal on Optimization* **11**, 611–646.

[23] D. C. Sorensen (1982) Newton's method with a model trust region modification, *SIAM Journal on Numerical Analysis* **19**, 409–426.

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