Globally Convergent DC Trust-Region Methods

Le Thi Hoai An* Huynh Van Ngai † Pham Dinh Tao ‡ A. Ismael F. Vaz§
L. N. Vicente¶

November 12, 2013

Abstract

In this paper, we investigate the use of DC (Difference of Convex functions) models and algorithms in the application of trust-region methods to the solution of a class of nonlinear optimization problems where the constrained set is closed and convex (and, from a practical point of view, where projecting onto the feasible region is computationally affordable). We consider DC local models for the quadratic model of the objective function used to compute the trust-region step, and apply a primal-dual subgradient method to the solution of the corresponding trust-region subproblems.

One is able to prove that the resulting scheme is globally convergent for first-order stationary points. The theory requires the use of exact second-order derivatives but, in turn, the computation of the trust-region step asks only for one projection onto the feasible region (in comparison to the calculation of the generalized Cauchy point which may require more).

The numerical efficiency and robustness of the proposed new scheme when applied to bound-constrained problems is measured by comparing its performance against some of the current state-of-the-art nonlinear programming solvers on a vast collection of test problems.

Keywords: Trust-region methods, DC algorithm, global convergence, bound constraints.

1 Introduction

Consider the constrained nonlinear programming problem

$$\min \ f(x) \quad \text{subject to} \quad x \in C, \tag{1}$$

*LITA, UFR MIM, Université de Lorraine, Ile du Saulcy 57045 Metz Cedex 01, France (hoai-an.le-thi@univ-lorraine.fr).

†Department of Mathematics, University of Quynhon, 170 An Duong Vuong, Quy Nhon, Vietnam.

‡Laboratory of Mathematics, National Institute for Applied Sciences-Rouen, 685 Avenue de l'Université BP8,

§Department of Production and Systems, University of Minho, Campus de Gualtar, 4710-057, Portugal (aivaz@dps.uminho.pt). Support for this author was provided by Algoritmi Research Center and by FCT under the grants PTDC/MAT/116736/2010 and PEst-C/EEI/UI0319/2011.

¶CMUC, Department of Mathematics, University of Coimbra, 3001-501 Coimbra, Portugal (lnv@mat.uc.pt). Support for this research was provided by FCT under grants PTDC/MAT/116736/2010 and PEst-C/MAT/UI0324/2011.

where $C \subseteq \mathbb{R}^n$ is a nonempty closed convex set and $f : \mathbb{R}^n \to \mathbb{R}$ is a twice continuously differentiable function. We have in mind a constraint set C over which projections are computationally affordable (like a set defined by bounds on the variables or other simpler settings such as the one considered in [9]). However, the algorithms and theory proposed in this paper apply to any closed convex set C.

Trust-region methods are widely acknowledged to be among the most efficient and robust methods for solving nonlinear optimization problems (see [8, 26]). A trust-region step results from the approximate solution of the trust-region subproblem, where a quadratic model of f is minimized over a trust-region ball of pre-specified size, possibly intersected with the feasible region C in the constrained case. When constraints of the form $x \in C$ are of polyhedral type, they can be naturally added to the trust-region subproblem (which would then consist of a quadratic program if the norm used in the trust-region ball is the ℓ_∞ one). Most trust-region methods compute the trust-region step in a way that the decrease produced in the quadratic model is a fraction of what is obtained by the so-called generalized Cauchy point, computed by determining the gradient-projected path (see [8, Chapter 12]).

The purpose of this paper is to integrate the DC Algorithm (DCA) in a trust-region framework for the solution of problem (1). DCA is a primal-dual subgradient method designed for solving a general DC program, i.e., an optimization problem where one minimizes the difference of convex functions on the whole space. Note that minimizing a DC function over a convext set C can be restated as a general DC program by using the indicator function χ_C of C. We apply DCA to the approximate solution of the trust-region subproblems, exploring specific DC decompositions of the quadratic models. The overall approach is shown to be globally convergent to first-order critical points (when the second-order information used in the quadratic model DC decompositions is exact). We will see that the theory requires only one DCA iteration to solve the trust-region subproblem, which amounts to only one projection onto the feasible region (and here we recall that the computation of the generalized Cauchy point may take more than one projection).

Our numerical experiments are focused entirely on the solution of bound-constrained problems. The numerical tests reported in this paper showed us that a few (cheap) DCA steps suffice to compute decently accurate trust-region steps, resulting in an efficient and reasonably robust algorithm. The minimization of a nonlinear function subject to bounds on the variables has been the subject of intense previous work, along many possible avenues. Major classes of algorithms for bound-constrained problems include the ones based on: active or ϵ-active set methods (see, e.g., [1, 13, 32] and more recently [18] for a short review on active set methods); trust-region methods (see, e.g., [6, 7, 14, 22, 24]); interior-point methods (see, e.g., [5, 10, 19]); line-search projected gradient methods (see, e.g., [2] and the references therein; see also [3, 25, 35] for a limited memory BFGS method); and filter type methods (see [31]). The approach proposed and analyzed in this paper belongs to the trust-region class but also shares the flavor of projected gradient methods.

We organize our contribution in the following way. In Section 2 we provide some background on the DC Algorithm. Our DC trust-region method is introduced and analyzed in Section 3. The two following sections are devoted to present our numerical findings. First we provide in Section 4 practical details of the implementation of the DC trust-region method, as well as information on how the numerical experiments were done and compared. The numerical results are then presented and commented on in Section 5. Some final conclusions are reported in Section 6. The norms and inner products used in the paper are the Euclidian ones.

2 DC programming, algorithm, and models

Let us start by recalling some basic notions from Convex Analysis and Nonsmooth Calculus which will be needed afterwards (see [4, 29, 30]). In the sequel, the space $\mathbb{R}^n$ is equipped with the Euclidean inner product $\langle \cdot, \cdot \rangle$. Let $\Gamma_0(\mathbb{R}^n)$ be the 'convex cone' of all the lower semicontinuous proper (i.e., not identically equal to $+\infty$) convex functions defined on $\mathbb{R}^n$ and taking values in $\mathbb{R} \cup \{+\infty\}$.

For $g \in \Gamma_0(\mathbb{R}^n)$, the subdifferential $\partial g(z)$ of g at a point z in its effective domain $\{z \in \mathbb{R}^n : g(z) < +\infty\}$ is defined by

$$\partial g(z) \; = \; \{w \in \mathbb{R}^n : \langle w, d \rangle \leq g(z+d) - g(z), \; \forall d \in \mathbb{R}^n\}$$

(by convention $\partial g(z) = \emptyset$ if z is not in the effective domain of g). The indicator function χ_C of a nonempty set C is defined by $\chi_C(z) = 0$ if $z \in C$, $+\infty$ otherwise. The normal cone $N(C, z)$ of a nonempty, closed, convex set C at $z \in C$ (the polar of the tangent cone) coincides with $\partial \chi_C(z)$, i.e.,

$$N(C, z) \; = \; \{u \in \mathbb{R}^n : \; \langle u, w - z \rangle \leq 0, \;\; \forall w \in C\} \; = \; \partial \chi_C(z).$$

For $\varphi \in \Gamma_0(\mathbb{R}^n)$, its conjugate φ^* is defined by

$$\varphi^*(w) \; = \; \sup\{\langle z, w \rangle - \varphi(z) : \; z \in \mathbb{R}^n\}$$

and it holds $\varphi^* \in \Gamma_0(\mathbb{R}^n)$ and $(\varphi^*)^* = \varphi$. The latter relation provides the crucial characterization of $\varphi \in \Gamma_0(\mathbb{R}^n)$ as a pointwise supremum of a collection of its affine minorants:

$$\varphi(z) \; = \; \sup\{\langle z, w \rangle - \varphi^*(w) : \; w \in \mathbb{R}^n\}. \tag{2}$$

A standard DC program is of the form (with the usual convention $(+\infty) - (+\infty) = +\infty$)

$$\inf\{f(z) := g(z) - h(z) : \; z \in \mathbb{R}^n\}, \tag{3}$$

where $g, h \in \Gamma_0(\mathbb{R}^n)$. DC programming deals with the vector space $DC(\mathbb{R}^n) = \Gamma_0(\mathbb{R}^n) - \Gamma_0(\mathbb{R}^n)$. Such a function f is called a DC function, and $g - h$ a DC decomposition of f, while the convex functions g and h are the DC components of f. Note that minimizing a DC function $f = g - h$ on a nonempty closed convex C set can be recast into the standard form (3) by changing g to $g + \chi_C$.

Using (2), the DC duality [21, 27, 28] associates a primal DC program with its dual defined as

$$\inf\{h^*(w) - g^*(w) : \; w \in \mathbb{R}^n\},$$

which is also a DC program with the same optimal value.

The DC Algorithm (DCA) is based on local optimality and DC duality, and has been introduced by Pham Dinh Tao in 1986 and extensively developed by Le Thi Hoai An and Pham Dinh Tao since 1994 (see [20, 21, 27, 28], and the references therein), being successfully applied to a number of classes of problems, including large-scale instances. DCA constructs two sequences $\{z^l\}$ and $\{w^l\}$ (of trial solutions of the primal and dual programs, respectively) which are improved at each iteration such that: (i) the sequences $\{g(z^l) - h(z^l)\}$ and $\{h^*(w^l) - g^*(w^l)\}$ are decreasing; (ii) their corresponding limit points z^∞ and w^∞ satisfy local optimality conditions, respectively for the primal and the dual. DCA is a descent method, without line search but globally convergent.

Algorithm 2.1 (DC Algorithm (DCA)) Initialization *Choose* $z^0 \in \mathbb{R}^n$.

For $l = 0, 1, \ldots$

1. *Compute* $w^l \in \partial h(z^l)$.
2. *Compute* $z^{l+1} \in \partial g^*(w^l)$, *i.e.,* z^{l+1} *is a solution of the convex program*

$$\min\{g(z) - \langle z, w^l\rangle \,:\, z \in \mathbb{R}^n\}.$$

If some stopping criterion is met, then stop, otherwise go to Step 1.

Output *Return* z^{l+1} *and* $g(z^{l+1}) - h(z^{l+1})$ *as the best known approximate solution and objective function value, respectively.*

The type of algorithms for solving problem (1) of interest to us in this paper are based on the iterative minimization of quadratic models on the intersection of C with a trust region, and for this purpose we want to use DC programming. Note that when the set C is defined by bounds on the variables and we choose the ℓ_∞-norm for the trust region, the resulting trust-region subproblems will consist of minimizing a quadratic function subject to box constraints.

In the sequel, the closed ball with center $x \in \mathbb{R}^n$ and radius $\varepsilon > 0$ is denoted by $B(x, \varepsilon)$.

Given $x \in \mathbb{R}^n$, we form a Taylor quadratic model of f around this point

$$m(x+p, x) \;=\; f(x) + \langle \nabla f(x), p\rangle + \frac{1}{2}\langle p, \nabla^2 f(x) p\rangle.$$

Note that when $\nabla^2 f$ is Lipschitz continuous with constant $\kappa > 0$ on $B(x, \Delta)$, one has

$$|f(x+p) - m(x+p, x)| \;\le\; \tfrac{\kappa}{6}\Delta^3, \tag{4}$$

for all $p \in B(0, \Delta)$.

The DC decomposition of $m(x+p, x)$ of most interest to us is

$$m(x+p, x) \;=\; m_g(x+p, x) - m_h(x+p, x),$$

where

$$m_g(x+p, x) \;=\; \frac{\rho_x}{2}\|p\|^2 + \chi_D(p) \quad \text{and} \quad m_h(x+p, x) \;=\; \frac{\rho_x}{2}\|p\|^2 - m(x+p, x),$$

$\rho_x = \|\nabla^2 m(x+p, x)\| = \|\nabla^2 f(x)\|$, and $D = (C - \{x\}) \cap B(0, \Delta)$ is the intersection of C (shifted by x) with the trust region $B(0, \Delta)$.

3 The DC trust-region method

At the iteration k, a step p_k is computed by approximately solving the trust-region subproblem

$$\min \; m(x_k + p, x_k) \quad \text{subject to} \quad p \in D_k = (C - \{x_k\}) \cap B(0, \Delta_k), \tag{5}$$

using the DCA (Algorithm 2.1) and the DC decomposition

$$m(x_k + p, x_k) \;=\; \left(\frac{\rho_{x_k}}{2}\|p\|^2 + \chi_{D_k}(p)\right) - \left(\frac{\rho_{x_k}}{2}\|p\|^2 - m(x_k + p, x_k)\right),$$

with $\rho_{x_k} = \|\nabla^2 f(x_k)\| + \epsilon$, where ϵ is a small positive quantity added to guarantee that ρ_{x_k} stays uniformly bounded away from zero.

The following algorithm summarizes our trust-region method using DCA for the trust-region subproblem minimization. The notation $P_W(z) = \arg\min_{w\in W} \|w - z\|$ denotes the projection of z onto a closed, convex set W. The algorithm is written without a stopping criterion to generate an infinite sequence of iterates for the subsequent analysis.

Algorithm 3.1 (DC trust-region algorithm)

Step 0 (initialization):
Choose an initial point $x_0 \in C$ and an initial trust-region radius $\Delta_0 > 0$. Select a positive integer l_0. Choose constants η_1, γ_1, $\gamma_2 \in (0, 1)$. Start with $k = 0$ and set $p_{-1} = 0$.

Step 1 (step calculation using DCA for subproblem):
Obtain p_k, with $\|p_k\| \leq \Delta_k$, by using DCA to approximately solve the trust-region subproblem (5), as follows:

Set $p_k^0 = P_{D_k}(p_{k-1})$.

For $l = 0, 1, \ldots, l_0 - 1$

1.
2. *Compute $q_k^l = \rho_{x_k} p_k^l - \nabla m(x_k + p_k^l, x_k)$.*
3. *Set $p_k^{l+1} = P_{D_k}(q_k^l/\rho_{x_k})$.*

Set $p_k = p_k^{l_0}$.

Step 2 (acceptance of trial point):
Compute $f(x_k + p_k)$ and define

$$\tau_k = \frac{f(x_k) - f(x_k + p_k)}{m(x_k, x_k) - m(x_k + p_k, x_k)}.$$

If $\tau_k \geq \eta_1$, then $x_{k+1} = x_k + p_k$. Otherwise define $x_{k+1} = x_k$.

Step 3 (trust-region radius update):
If $\tau_k \geq \eta_1$ then $\Delta_{k+1} \in [\Delta_k, +\infty)$, otherwise $\Delta_{k+1} \in [\gamma_1\Delta_k, \gamma_2\Delta_k]$.

*Increment k by 1 and go to **Step 1**.*

In fact, regarding Step 1 where DCA (Algorithm 2.1) is applied to solve the trust-region subproblem (5), we point out that q_k^l corresponds to $w^l = \nabla h(z^l)$ in Algorithm 2.1 and that p_k^{l+1}, the solution of $\min\{m_g(x_k + p) - \langle p, q_k^l\rangle : p \in \mathbb{R}^n\}$, corresponds to z^{l+1} in Algorithm 2.1.

Note that the minimal effort per iteration in this algorithm (when $l_0 = 1$) amounts to one projection, and that this compares to the computation of a generalized Cauchy point (see [8, Algorithm 12.2.2]) for trust-region methods when applied to general convex constrained problems which may take more than one projection.

Another point is that when $p_k^0 = 0$ and $l_0 = 1$, a main step of Algorithm 3.1 resembles a DCA step applied to the original smooth problem of the form (1), since $q_k^0 = \rho_{x_k} p_k^0 - \nabla m(x_k + p_k^0, x_k) = -\nabla f(x_k)$. However the choice of ρ_{x_k} is local and it does not render a true global DC decomposition for the original problem.

We are now in a position to show global convergence to first-order stationary point.

Theorem 3.1 *Let $\{x_k\}$ be a sequence generated by Algorithm 3.1 applied to a twice continuously differentiable function f for which $\nabla^2 f$ is Lipschitz continuous on C. Then the sequence $\{f(x_k)\}$ is decreasing and $\lim_{k\to+\infty}\|x_{k+1}-x_k\| = 0$. Moreover, every limit point x_∞ of the sequence $\{x_k\}$ is a first-order critical point of problem (1), that is, $0 \in \nabla f(x_\infty) + N(C, x_\infty)$, where $N(C, x_\infty)$ stands for the normal cone of the convex set C at the point x_∞.*

Proof. Consider first the convex quadratic function in $p \in \mathbb{R}^n$ given by $m_h(x_k + p, x_k) = \frac{\rho_{x_k}}{2}\|p\|^2 - m(x_k + p, x_k)$. Hence, for every k and $l = 0, 1, \ldots, l_0 - 1$, one has directly from the first-order characterization of convexity of $m_h(x_k + p, x_k)$

$$\langle q_k^l, p_k^{l+1} - p_k^l \rangle \ \leq\ m_h(x_k + p_k^{l+1}, x_k) - m_h(x_k + p_k^l, x_k), \tag{6}$$

where $q_k^l = \nabla m_h(x_k + p_k^l, x_k) = \rho_{x_k} p_k^l - \nabla m(x_k + p_k^l, x_k)$. On the other hand, since $p_k^{l+1} = P_{D_k}(q_k^l/\rho_{x_k})$, from the definition of projection one has

$$\rho_{x_k}\langle q_k^l/\rho_{x_k} - p_k^{l+1}, p_k^l - p_k^{l+1}\rangle \ \leq\ 0,$$

which is equivalent to

$$\langle q_k^l, p_k^l - p_k^{l+1}\rangle \ \leq\ \frac{\rho_{x_k}}{2}\|p_k^l\|^2 - \frac{\rho_{x_k}}{2}\|p_k^{l+1}\|^2 - \frac{\rho_{x_k}}{2}\|p_k^l - p_k^{l+1}\|^2. \tag{7}$$

From inequalities (6) and (7), one then obtains

$$m(x_k + p_k^l, x_k) - m(x_k + p_k^{l+1}, x_k) \ \geq\ \frac{\rho_{x_k}}{2}\|p_k^l - p_k^{l+1}\|^2$$

and therefore

$$\begin{aligned} m(x_k, x_k) - m(x_k + p_k, x_k) \ &=\ \textstyle\sum_{l=0}^{l_0-1}[m(x_k + p_k^l, x_k) - m(x_k + p_k^{l+1}, x_k)] \\ &\geq\ \tfrac{\rho_{x_k}}{2}\textstyle\sum_{l=0}^{l_0-1}\|p_k^l - p_k^{l+1}\|^2 \ \geq\ \tfrac{\rho_{x_k}}{2l_0}\|p_k\|^2. \end{aligned} \tag{8}$$

Now denote by κ the Lipschitz constant of $\nabla^2 f$ on C. By the definition of τ_k and by applying a Taylor expansion [11, Lemma 4.1.14] (in the numerator below), one has

$$|\tau_k - 1| \ =\ \left|\frac{m(x_k + p_k, x_k) - f(x_k + p_k)}{m(x_k, x_k) - m(x_k + p_k, x_k)}\right| \ \leq\ \frac{(\kappa/6)\|p_k\|^3}{\rho_{x_k}\|p_k\|^2/2l_0} \ =\ \frac{\kappa l_0}{3\rho_{x_k}}\|p_k\|.$$

Thus, since $\rho_{x_k} \geq \epsilon$ and $\|p_k\| \leq \Delta_k$, if

$$\Delta_k \ \leq\ \frac{3(1-\eta_1)\epsilon}{\kappa l_0},$$

then the iteration is successful. One can conclude that there is an infinity of successful iterations. Moreover, from the trust-region update of the algorithm, one has that

$$\Delta_k \ \geq\ \Delta_{\min} \ =\ \frac{3(1-\eta_1)\epsilon\gamma_1}{\kappa l_0} \quad \text{for all } k.$$

Furthermore, ignoring the unsuccessful iterations where there is no displacement, one obtains

$$f(x_k) - f(x_{k+1}) \ =\ \tau_k\left[m(x_k, x_k) - m(x_{k+1}, x_k)\right] \ \geq\ \frac{\rho_{x_k}\eta_1}{2l_0}\|x_k - x_{k+1}\|^2. \tag{9}$$

Consequently, $f(x_k)$ is a monotonically decreasing sequence. Since f is bounded from below, $f(x_k)$ converges. As a result one has that $\lim_{k\to+\infty}\|x_{k+1}-x_k\|=0$.

Let x_∞ be a limit point of the sequence $\{x_k\}$, say, $\lim_{i\to+\infty} x_{k_i}=x_\infty$ for some subsequence $\{x_{k_i}\}$ of $\{x_k\}$. Since we proved above that there is an infinity of successful iterations, for all $i=1,2,\ldots$ there exists an index $j_i\geq 1$ such that

$$x_{k_i} \;=\; x_{k_i+1} \;=\; \cdots \;=\; x_{k_i+j_i-1} \;\neq\; x_{k_i+j_i}.$$

One knows from (9) that $\lim_{i\to+\infty}\|p^{l_0}_{k_i+j_i-1}\| = \|p_{k_i+j_i-1}\|=0$, and by using this and taking limits in (8), we obtain $\lim_{i\to+\infty}\|p^1_{k_i+j_i-1}\|=0$. Since

$$\begin{aligned}&x_{k_i+j_i-1}+p^1_{k_i+j_i-1}\\ &= P_{C\cap B(x_{k_i+j_i-1},\Delta_{k_i+j_i-1})}\left(x_{k_i+j_i-1}-\nabla f(x_{k_i+j_i-1})/\rho_{x_{k_i+j_i-1}}-\nabla^2 f(x_{k_i})p^1_{k_i+j_i-1}/\rho_{x_{k_i}}\right),\end{aligned}$$

we have

$$\langle -\nabla f(x_{k_i})/\rho_{x_{k_i}}-\nabla^2 f(x_{k_i})p^1_{k_i+j_i-1}/\rho_{x_{k_i}}-p^1_{k_i+j_i-1}, x-x_{k_i}-p^1_{k_i+j_i-1}\rangle \;\leq\; 0$$

for all $x\in C\cap B(x_{k_i},\Delta_{k_i+j_i-1})$. By taking the limits $x_{k_i}\to x_\infty$, $p^1_{k_i+j_i-1}\to 0$, and $\rho_{x_{k_i}}\to \|\nabla^2 f(x_\infty)\|+\epsilon$, one obtains

$$\langle -\nabla f(x_\infty)/(\|\nabla^2 f(x_\infty)\|+\epsilon), x-x_\infty\rangle \;\leq\; 0.$$

Recalling that $\Delta_k\geq\Delta_{\min}>0$ for all k, one obtains the desired conclusion $-\nabla f(x_\infty)\in N(C,x_\infty)$. □

Interestingly, it is possible to replace τ_k by

$$\tau_k^{new} \;=\; \frac{2l_0(f(x_k)-f(x_k+p_k))}{\rho_{x_k}\|p_k\|^2} \tag{10}$$

and obtain a similar result.

Corollary 3.1 *Let $\{x_k\}$ be a sequence generated by Algorithm 3.1, under the modification* (10)*, applied to a twice continuously differentiable function f for which $\nabla^2 f$ is Lipschitz continuous on C. Then the sequence $\{f(x_k)\}$ is decreasing and $\lim_{k\to+\infty}\|x_{k+1}-x_k\|=0$. Moreover, every limit point x_∞ of the sequence $\{x_k\}$ is a first-order critical point of problem $(\mathcal{P})$, that is, $0\in\nabla f(x_\infty)+N(C,x_\infty)$.*

Proof. From (8) one obtains that $\tau_k\geq\eta$ implies $\tau_k^{new}\geq\eta$. Thus, if an iteration is successful for Algorithm 3.1 so it is for the modified version of the algorithm. The rest of the proof is exactly as in the one of Theorem 3.1. □

The search direction p_k could have also been computed by solving approximately the trust-region subproblem (5) using the DCA (Algorithm 2.1) and the DC decomposition

$$m(x_k+p,x_k) \;=\; \left(m(x_k+p,x_k)+\frac{\rho_{x_k}}{2}\|p\|^2+\chi_{D_k}(p)\right)-\left(\frac{\rho_{x_k}}{2}\|p\|^2\right),$$

with $\rho_{x_k}=\max\{-\lambda_{\min}(\nabla^2 f(x_k)),0\}+\epsilon$, where $\lambda_{\min}(\cdot)$ denotes the smallest eigenvalue of a matrix. The authors believe that it is possible to obtain the same convergent result for this decomposition as the one described in Theorem 3.1. However, each internal iteration of DCA would have then required the solution of an auxiliary problem of the form

$$\min\ m(x_k + p, x_k) + \frac{\rho_{x_k}}{2}\|p\|^2 - \langle p, q_k^l\rangle \quad \text{subject to} \quad p \in D_k,$$

which would have been more expensive when compared to what happens in Algorithm 3.1, where the bulk of the work per one internal iteration of DCA amounts to one projection onto D_k.

4 Implementation issues, test problems, and profiles

4.1 Implementation issues

To provide an assessment of the proposed methodology we developed an implementation for Algorithm 3.1, called `TRDC` (Trust Region Difference of Convex). As already mentioned in the introduction, our implementation only addresses bound-constrained problems, i.e., problems of the form (1) where $C = \{x \in \mathbb{R}^n : \ell \le x \le u\}$, with $\ell \in (\mathbb{R} \cup \{-\infty\})^n$ and $u \in (\mathbb{R} \cup \{+\infty\})^n$. To make projections onto $(C - \{x_k\}) \cap B(0, \Delta_k)$ fast, see (5), we considered $B(0, \Delta_k)$ defined using the ℓ_∞-norm. Since the solvers that we are using later for the numerical results are implemented in `C++` (`IPOPT`) and Fortran (`Lancelot B` and `TRON`), and in order to provide a fair comparison, our implementation of `TRDC` was made in `C`.

While Algorithm 3.1 requests a positive integer l_0 (the number of internal DCA iterations), performing more internal iterations than needed to solve the trust-region subproblem (5) will lead to inefficiency. Also, considering $\rho_{x_k} = \|\nabla^2 f(x_k)\| + \epsilon$ may also lead to a high number of DCA internal iterations. Therefore, we consider an adaptive strategy for updating ρ_{x_k}, making it also dependent on the DCA internal loop counter l. Thus, ρ_{x_k} will be hereafter denoted by $\rho_{x_k}^l$. We start with a smaller value $\rho_{x_k}^0$ (set to $2^{-2}\left(\|\nabla^2 f(x_k)\| + \epsilon\right)$, $\epsilon = 0.1$, in our implementation), and multiply it by a factor of $\rho_{factor} = 2$ in each inner iteration l. We then stop the DCA internal loop if $m(x_k, x_k) - m(x_k + p_k^{l+1}, x_k) \ge C\|p_k^{l+1}\|^2$, with $C = 10^3$. Such a stopping criterion is totally consistent with our theory, even when $\rho_{x_k}^l$ is still below $\|\nabla^2 f(x_k)\| + \epsilon$, since we can see from the subsequent use of the lower bound (8) that all it is required is a reduction in the predicted decrease of the order of the square of the step (being irrelevant the size of the constant multiplying it). Finally, a maximum number of $l_0 = 300$ DCA inner iterations is also considered.

As in `TRON`, which is also a bound-constrained type solver, we stop the external iterations, declaring success, whenever the absolute error in the objective function is small, $|f(x_k) - f(x_k + p_k)| < \epsilon_{tol}$, and the predicted reduction is small, $m(x_k, x_k) - m(x_k + p_k, x_k) < \epsilon_{tol}$, with ϵ_{tol} set to 10^{-12} as in `TRON`. A run of `TRDC` is stopped unsuccessfully if it exceeds a maximum number of external iterations ($maxiter$), a maximum of total internal DCA iterations ($maxiterDCA$), or a maximum of objective function evaluations ($maxfeval$), with $maxiterDCA = 10^7$, $maxfeval = 1000$, and $maxiter = 1000$.

To improve numerical performance, and as in `IPOPT`, we considered instead a scaled objective function f^*, given by $f^*(x) = \zeta f(x)$, with

$$\zeta \ = \ \min\left(1, \frac{100}{\|\nabla f(x_0)\|}\right),$$

where x_0 is the projection onto the feasible region of the user provided initial guess (e.g., given by CUTEr [15]). The scaling parameter ζ is computed at the algorithm initialization and kept fixed for the remaining procedure. When x_0 is not provided, we compute a feasible initial guess in the following componentwise fashion: the middle value of the bounds when both are finite, the finite bound when one of the bounds is finite, or 0 whenever the variable is free.

A final implementation issue is related to the update of the trust-region radius Δ_k, described in `Step 3` of Algorithm 3.1. We provide the details of the updating scheme for Δ_k in the following algorithm.

Trust-region radius update

- If $\tau_k > \eta_3$,
  - then increase the trust-region radius by setting $\Delta_{k+1} = \min(\bar{\gamma}_3 \Delta_k, 1000)$,
  - otherwise, if $\tau_k < \eta_1$
    - then set $\Delta_{k+1} = \bar{\gamma}_1 \Delta_k$
    - otherwise if $\tau_k < \eta_2$,
      - then set $\Delta_{k+1} = \bar{\gamma}_2 \Delta_k$
      - otherwise set $\Delta_{k+1} = \Delta_k$.

By taking $0 < \eta_1 \leq \eta_2 \leq \eta_3 < 1$ and $0 < \bar{\gamma}_1 \leq \bar{\gamma}_2 < 1$, $\bar{\gamma}_3 \geq 1$, this scheme satisfies the conditions required in `Step 3` of Algorithm 3.1. In practice, we started with $\Delta_0 = 1$ and used $\eta_1 = 10^{-3}$, $\eta_2 = 0.25$, $\eta_3 = 0.75$, $\bar{\gamma}_1 = 0.5$, $\bar{\gamma}_2 = 0.5$, and $\bar{\gamma}_3 = 2$.

4.2 Test problems

In order to insure a proper comparison of the implemented solver with state-of-the-art optimization solvers, we decided to consider the CUTEr [15] test problems collection. From the complete test set there available, we selected all the unconstrained and bound-constrained problems, resulting in the 271 test problems reported in Table 1.

4.3 Profiles

Using a large number of test problems demands for an aggregated way to show the numerical results. For a better visualization and brevity in the presentation of the numerical results, we are providing performance profiles obtained by using the procedure described in [12]. We consider also the modification made in [33] for the case where the metric used for performance does not always return a strictly positive value, as required in the original performance profiles. The major advantage of performance profiles is that they can be presented in one figure, by plotting, for the different solvers, a cumulative distribution function $\upsilon(\pi)$ representing a performance ratio.

The performance ratio is defined by setting $r_{p,s} = \frac{t_{p,s}}{\min\{t_{p,z} : z \in \mathcal{S}\}}$, $p \in \mathcal{P}$, $s \in \mathcal{S}$, where $\mathcal{P}$ is the test set, $\mathcal{S}$ is the set of solvers, and $t_{p,s}$ is a measure of performance of the application of solver s on test problem p. Then, one defines $\upsilon_s(\pi) = \frac{1}{|\mathcal{P}|}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \pi\}$, where $|\mathcal{P}|$ is the number of test problems. The value of $\upsilon_s(1)$ is then the percentage of times that the solver s wins over the remaining ones (or ties the best solver). If we are only interested in determining which solver is the best (in the sense that wins the most), we compare the values of $\upsilon_s(1)$ for all the solvers. At the other end, $\upsilon_s(\pi)$ for large values of π indicates the percentage of problems solved successfully by solver s, and thus serves as a measure of robustness.

Problem	n	Problem	n	Problem	n	Problem	n	Problem	n
BQP1VAR	1	KOEBHELB	3	PALMER5A	8	EXPLIN	1200	SBRYBND	5000
AKIVA	2	MEYER3	3	PALMER5D	8	EXPLIN2	1200	SCHMVETT	5000
BEALE	2	PFIT1LS	3	PALMER5E	8	EXPQUAD	1200	SCOSINE	5000
BRKMCC	2	PFIT2LS	3	PALMER6C	8	LINVERSE	1999	SINQUAD	5000
BROWNBS	2	PFIT3LS	3	PALMER6E	8	EDENSCH	2000	SPARSINE	5000
CAMEL6	2	PFIT4LS	3	PALMER7C	8	RAYBENDL	2050	SROSENBR	5000
CLIFF	2	WEEDS	3	PALMER7E	8	RAYBENDS	2050	TESTQUAD	5000
CUBE	2	YFIT	3	PALMER8C	8	DIXMAANA	3000	TOINTGSS	5000
DENSCHNA	2	YFITU	3	PALMER8E	8	DIXMAANB	3000	TQUARTIC	5000
DENSCHNB	2	ALLINIT	4	S368	8	DIXMAANC	3000	TRIDIA	5000
DENSCHNC	2	ALLINITU	4	VIBRBEAM	8	DIXMAAND	3000	SCOND1LS	5002
DENSCHNF	2	BROWNDEN	4	PALMER5B	9	DIXMAANE	3000	BRATU1D	5003
DJTL	2	HATFLDA	4	SPARSQUR	9	DIXMAANF	3000	CLPLATEA	5041
EXPFIT	2	HATFLDB	4	SPECAN	9	DIXMAANG	3000	CLPLATEB	5041
HAIRY	2	HIMMELBF	4	HILBERTB	10	DIXMAANH	3000	CLPLATEC	5041
HILBERTA	2	HS38	4	HS110	10	DIXMAANI	3000	ODC	5184
HIMMELBB	2	KOWOSB	4	OSCIPATH	10	DIXMAANJ	3000	SSC	5184
HIMMELBG	2	PALMER1	4	OSBORNEB	11	DIXMAANL	3000	MINSURFO	5306
HIMMELBH	2	PALMER1B	4	WATSON	12	CHAINWOO	4000	NOBNDTOR	5476
HIMMELP1	2	PALMER2	4	DIXMAANK	15	WOODS	4000	TORSION1	5476
HS1	2	PALMER2B	4	HATFLDC	25	DRCAV1LQ	4489	TORSION2	5476
HS2	2	PALMER3	4	3PK	30	DRCAV2LQ	4489	TORSION3	5476
HS3	2	PALMER3B	4	BQPGABIM	50	DRCAV3LQ	4489	TORSION4	5476
HS3MOD	2	PALMER4	4	BQPGASIM	50	SPMSRTLS	4999	TORSION5	5476
HS4	2	PALMER4B	4	CHNROSNB	50	ARWHEAD	5000	TORSION6	5476
HS5	2	PSPDOC	4	ERRINROS	50	BDEXP	5000	TORSIONA	5476
HUMPS	2	HS45	5	TOINTGOR	50	BDQRTIC	5000	TORSIONB	5476
JENSMP	2	OSBORNEA	5	TOINTPSP	50	BIGGSB1	5000	TORSIONC	5476
LOGHAIRY	2	BIGGS3	6	TOINTQOR	50	BROYDN7D	5000	TORSIOND	5476
LOGROS	2	BIGGS5	6	VAREIGVL	50	BRYBND	5000	TORSIONE	5476
MARATOSB	2	BIGGS6	6	DECONVB	61	CHENHARK	5000	TORSIONF	5476
MDHOLE	2	HART6	6	DECONVU	61	CRAGGLVY	5000	FMINSRF2	5625
MEXHAT	2	HEART6LS	6	MINSURF	64	DQDRTIC	5000	LMINSURF	5625
ROSENBR	2	PALMER1A	6	HYDC20LS	99	DQRTIC	5000	NLMSURF	5625
S308	2	PALMER2A	6	CHEBYQAD	100	ENGVAL1	5000	GRIDGENA	6218
SIM2BQP	2	PALMER3A	6	MANCINO	100	FLETCBV2	5000	COSINE	10000
SIMBQP	2	PALMER4A	6	SENSORS	100	FLETCBV3	5000	CURLY10	10000
SINEVAL	2	PALMER5C	6	ARGLINA	200	FLETCHBV	5000	CURLY20	10000
SISSER	2	PALMER6A	6	ARGLINB	200	FREUROTH	5000	CVXBQP1	10000
SNAIL	2	PALMER7A	6	ARGLINC	200	GENHUMPS	5000	DIXON3DQ	10000
ZANGWIL2	2	PALMER8A	6	BROWNAL	200	INDEF	5000	JNLBRNG1	10000
BARD	3	PALMER1D	7	PENALTY2	200	LIARWHD	5000	JNLBRNG2	10000
BOX2	3	AIRCRFTB	8	VARDIM	200	MCCORMCK	5000	JNLBRNGA	10000
BOX3	3	HEART8LS	8	HADAMALS	400	MOREBV	5000	JNLBRNGB	10000
DENSCHND	3	MAXLIKA	8	GENROSE	500	NONCVXU2	5000	NCVXBQP1	10000
DENSCHNE	3	OSLBQP	8	PROBPENL	500	NONCVXUN	5000	NCVXBQP2	10000
EG1	3	PALMER1C	8	QR3DLS	610	NONDIA	5000	NCVXBQP3	10000
ENGVAL2	3	PALMER1E	8	EG2	1000	NONDQUAR	5000	OBSTCLAE	10000
GROWTHLS	3	PALMER2C	8	EXTROSNB	1000	NONSCOMP	5000	OBSTCLAL	10000
GULF	3	PALMER2E	8	FLETCHCR	1000	PENTDI	5000	OBSTCLBL	10000
HATFLDD	3	PALMER3C	8	PENALTY1	1000	POWELLSG	5000	OBSTCLBM	10000
HATFLDE	3	PALMER3E	8	SINEALI	1000	QRTQUAD	5000	OBSTCLBU	10000
HATFLDFL	3	PALMER4C	8	MSQRTALS	1024	QUARTC	5000	SCURLY10	10000
HELIX	3	PALMER4E	8	MSQRTBLS	1024	QUDLIN	5000	SCURLY20	10000
HS25	3								

Table 1: CUTEr test problems used in the numerical results.

Clearly, when for a certain problem $p \in \mathcal{P}$ one has $\min\{t_{p,z} : z \in \mathcal{S}\} \leq 0$, the value $r_{p,s}$ becomes meaningless or undefined for all $s \in \mathcal{S}$. We considered two possibilities to overcome this problem in our numerical results. One is to simply exclude all problems where such a situation happens, reducing the number of test problems to be included and then using the original performance profiles [12]. The second one is to keep all problems and to choose a way to deal with problems where $t_{p,s} \leq 0$ happens for at least one solver s. We considered $r_{p,s} = t_{p,s} + 1 - \min\{t_{p,z} : z \in \mathcal{S}\}$ whenever $\min\{t_{p,z} : z \in \mathcal{S}\} < 0.0001$ to overcome the possibility of $r_{p,s}$ being meaningless or undefined (see [33] for further details).

5 Numerical results

Since our proposed method uses second order derivatives we decided to compare it against `TRON` [23, 24], `IPOPT` [34], and `Lancelot B` [16] (available under the GALAHAD library [17]), which represent well the state-of-the-art optimization solvers where second order derivatives are used. `TRON` was specially developed to address bound-constrained optimization problems, while `IPOPT` and `Lancelot B` can handle more general constrained optimization problems.

Since the computational effort made per iteration of `IPOPT`, `Lancelot B`, `TRDC`, and `TRON` is substantial different, we chose to compare the overall CPU time taken by the solvers. As it was said before, `TRDC` and `TRON` have similar stopping criteria, being the one for `TRON` slightly more advantageous for declaring success since it uses either the relative or the absolute error in function values. `IPOPT` and `Lancelot B` were run using a tolerance of 10^{-5} (the default value for `Lancelot B`) in their stopping criteria: `Lancelot B` uses the norm of the projected gradient while `IPOPT` uses the maximum between a scaled norm of the gradient of the Lagrangian and the complementarity residual.

The numerical experiments were made in an Intel(R) Core(TM) Duo CPU computer, running at 2.66GHz, under a Linux operating system, using recent versions for all the solvers (`TRON` version 1.2, `IPOPT` version 3.10.1, and GALAHAD version dated of February 2011). The same exact CUTEr collection (version date `CUTEr: Mon Jan 8 15:36:20 EST 2007`) was used by the four solvers, and for each problem the same CUTEr initial point was considered. For each problem, a maximum running time of 3600 seconds was imposed to all solvers (i.e., a failure is declared for a solver on a problem if it is unable to provide an answer in less than 3600 seconds). The considered CPU time corresponds to the CPU time taken by the solver (excluding the CUTEr setup time) measured in seconds with two decimal places. Due to this limitation in measuring the solver CPU time, one can easily obtain $t_{p,s} = 0$ for many 'easy' problems.

A first set of performance profiles is provided in Figures 1 and 2, where all the test problems were considered and then the performance profiles from [33] were used. For a matter of visibility around $\upsilon_s(1)$ and along $\upsilon_s(\pi)$ for large values of π, we considered two subfigures in each figure. These profiles were depicted trusting the exit flag produced by each solver in order to check if success was attained on solving a given problem. From these performance profiles one can conclude that the `TRDC` solver is competitive in both efficiency and robustness, when compared to the other solvers, specially for problems with less than 2000 variables.

A closer look at these numerical results reveals that for problems `FLETCBV3`, `FLETCHBV`, `INDEF`, `QRTQUAD`, `SCURLY10`, and `SCURLY20` all the solvers were unable to converge. These problems seem to have an unbounded objective function and therefore were removed in the next round of profiles. Additionally, we list, on Table 2, the problems where a solver reported an unsuccessful

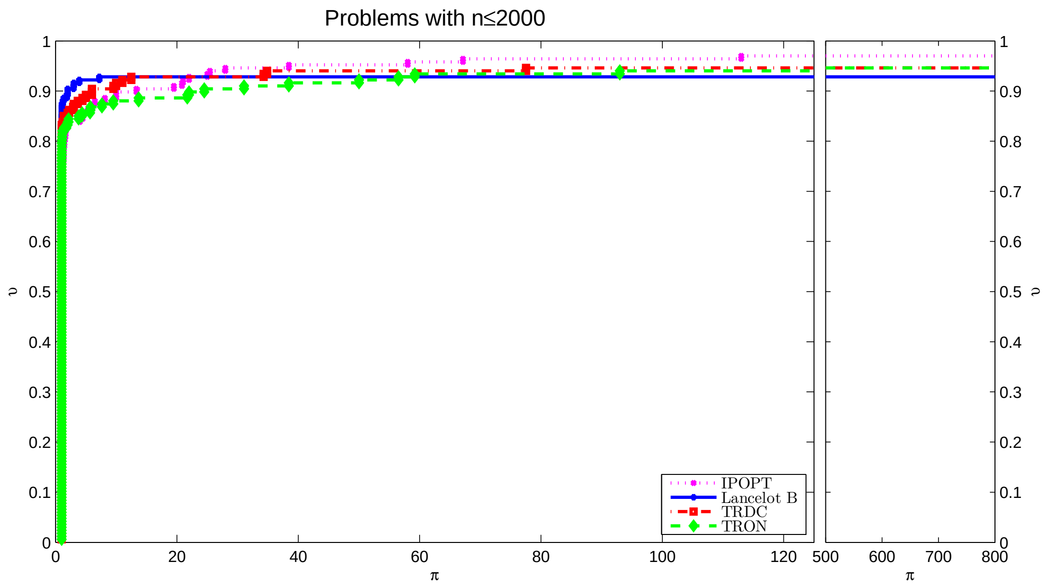

Figure 1: Performance profiles [33] for CPU time used by `IPOPT`, `Lancelot B`, `TRDC`, and `TRON`. Problems with $n \leq 2000$.

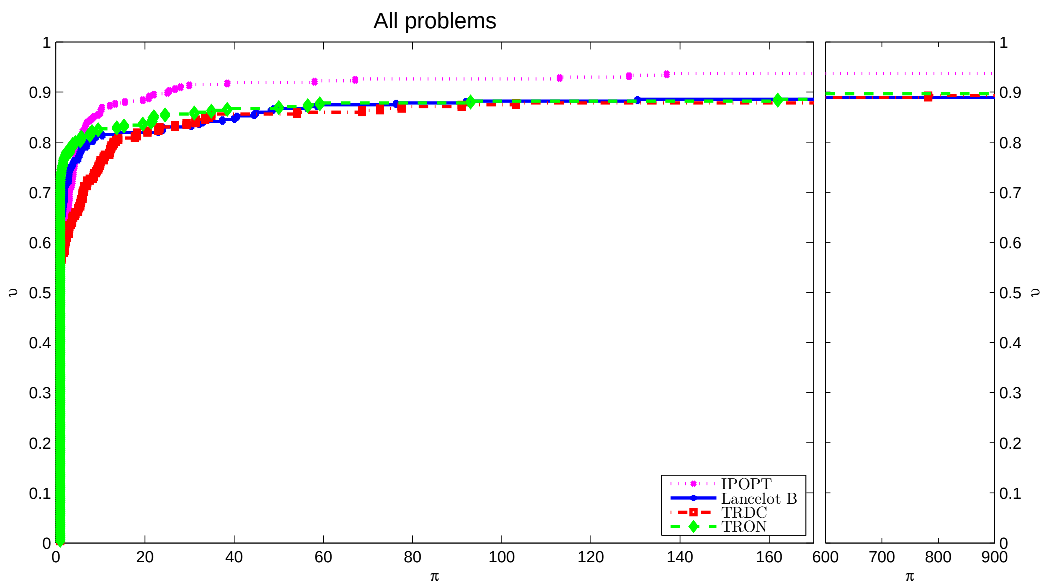

Figure 2: Performance profiles [33] for CPU time used by `IPOPT`, `Lancelot B`, `TRDC`, and `TRON`. All problems.

IPOPT	Lancelot B	TRDC
OSCIPATH	ARGLINB	CURLY20
PALMER5A	ARGLINC	EXPLIN
	BIGGSB1	EXPLIN2
	DJTL	EXPQUAD
	HEART6LS	FREUROTH
	LOGHAIRY	NCVXBQP1
	OSCIPATH	NCVXBQP2
	PALMER5A	NCVXBQP3
	PALMER7A	PENALTY2
		QUDLIN
		RAYBENDL
		RAYBENDS
		SINQUAD

Table 2: Test problems considered as successfully solved despite un unsuccessful exit flag.

exitflag, but the final objective function value is close to the one reported by the other solvers. For these problems we will now consider the run to be successful and use the corresponding CPU time. (To be more rigorous, suppose solver B solved problem p successfully. A solver A is considered to have solved problem p successfully, even when reporting an unsuccessful error flag, in the situations where either (i) $f_{p,A} \leq f_{p,B}$ or (ii) $\frac{|f_{p,A}-f_{p,B}|}{\max(1,\min(|f_{p,A}|,|f_{p,B}|))} \leq 10^{-3}$, where $f_{p,A}$ and $f_{p,B}$ represent the final objective function values of the two solvers for problem p.) We point out that `TRON` always reports a successful exit flag (equal to zero), i.e., the only unsuccessful cases are due to exceeding the 3600 seconds running time limit.

The new performance profiles for the restricted test set are depicted in Figures 3 and 4. The relative position of each solver is similar, but these new profiles reassure the robustness of the `TRDC` solver, as it was able to solve more than 90% of the problems.

We have also built other performance profiles, namely for the cases where the considered problems had dimensions $n \leq 10, 50, 100, 500, 1000, 3000, 5000$. We also plotted the performance profiles for each type of constraints available in the problems: unconstrained problems ($\ell = (-\infty)^n$ and $u = (+\infty)^n$), bound-constrained problems with at least one finite bound and one infinite bound, and bound-constrained problems with $\ell \in \mathbb{R}^n$ and $u \in \mathbb{R}^n$. Since we observed no major differences between these performance profiles and the ones reported before, we decided not to present them here for sake of brevity.

For a matter of completeness we also report our numerical findings using the original performance profiles [12]. In order to be able to plot these profiles we exclude from the test set all 'easy' problems for which $\min\{t_{p,z} : z \in \mathcal{S}\} = 0$ (i.e., all problems where at least one solver terminated using $t_{p,s} = 0$). This technique is the same as the one used in [18], where numerical results were restricted to problems with a running time of the fastest solver exceeding .01 seconds. These new performance profiles (for the restricted test set previously described) are presented in Figures 5 and 6. The number of problems is reduced from 167 to 30 (when $n \leq 2000$) and from 265 to 124 (all dimensions considered). While these new performance profiles are in accordance with the original ones in [12], the number of problems considered is considerably smaller and

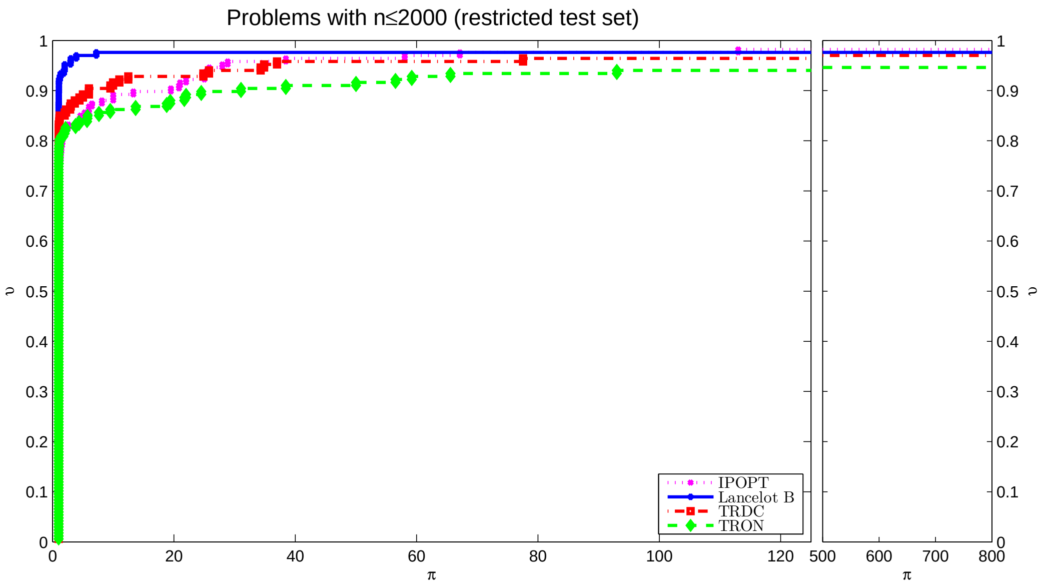

Figure 3: Performance profiles [33] for CPU time used by `IPOPT`, `Lancelot B`, `TRDC`, and `TRON`. Restricted test set, problems with $n \leq 2000$.

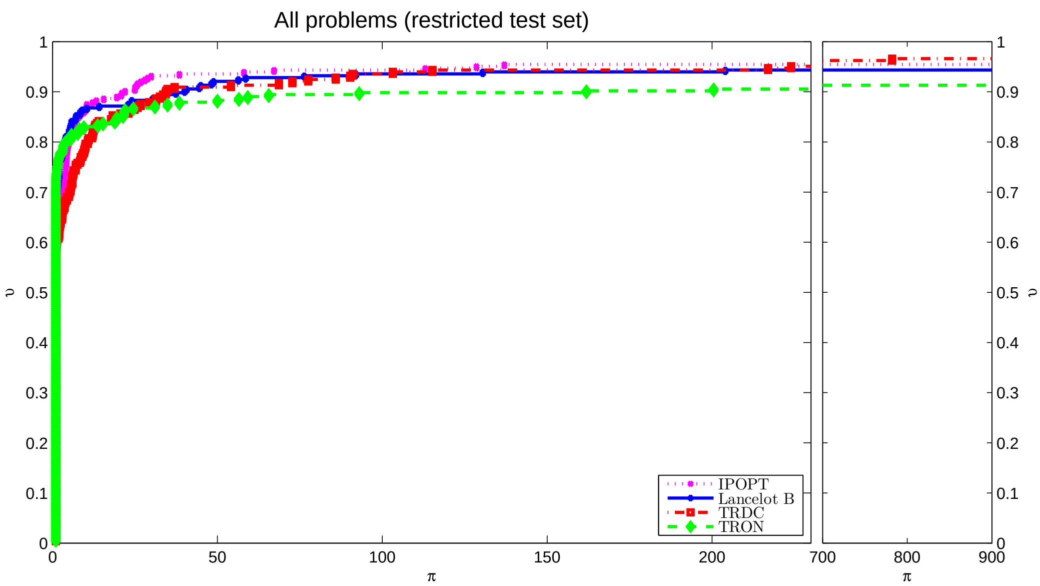

Figure 4: Performance profiles [33] for CPU time used by `IPOPT`, `Lancelot B`, `TRDC`, and `TRON`. All problems in the restricted test set.

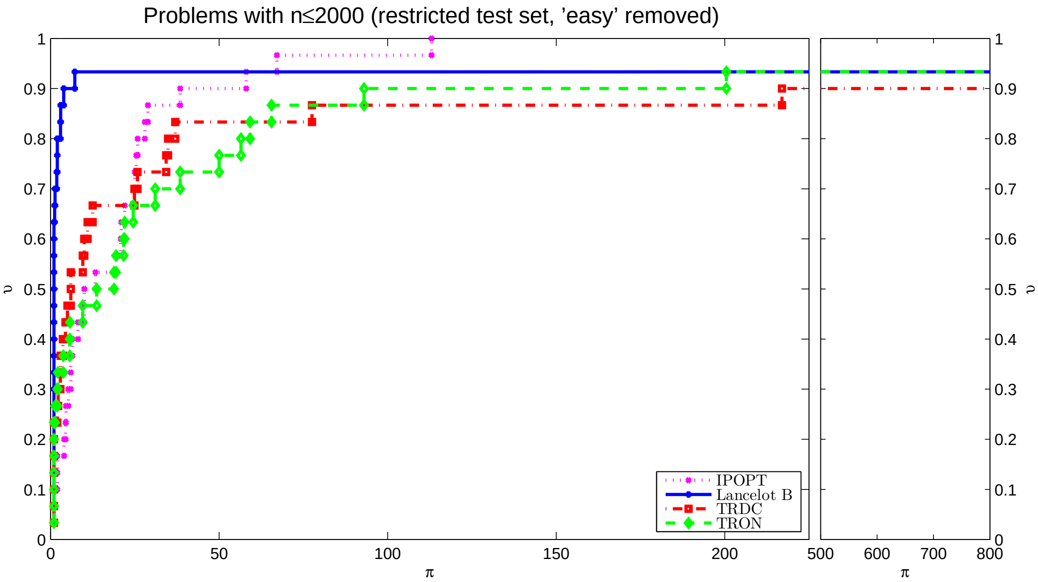

Figure 5: Performance profiles [12] for CPU time used by `IPOPT`, `Lancelot B`, `TRDC`, and `TRON`. Restricted test set, problems with $n \leq 2000$ except the 'easy' ones.

the best solvers are likely to be in disadvantage since problems where a solver attains $t_{p,s} = 0$ are removed from the analysis (disregardless of other solvers having or not $t_{p,s} \gg 0$).

From the depicted profiles for $n \leq 2000$, one can observe that `Lancelot B` is the most efficient solver, while `IPOPT`, `Lancelot B`, and `TRON` are the most robust. `TRDC` presents a similar performance in terms of efficiency and attains a robustness of about 90%. When all the problems in the restricted test set are considered we observe a slight advantage for the `TRDC` solver.

6 Conclusions

A trust-region type method has been proposed, analyzed, and implemented, involving a new minimum requirement, from the solution of the trust-region subproblems, for achieving global convergence to first-order stationary points. Such a requirement, different from Cauchy or generalized Cauchy points, is related to the application of a first step of a primal-dual subgradient method (the DC algorithm), and it necessarily involves the knowledge of second-order derivatives, although it only requires one projection onto the feasible set. One is able to prove, in a relatively short and clean argument, that all limit points of the sequence of iterates are first-order critical. The numerical experiments reported show that the new approach is competitive with state-of-the-art solvers for problems with bounds on the variables.

Acknowledgements

The authors thank the referees for their constructive comments and suggestions which led to an improved version of the paper.

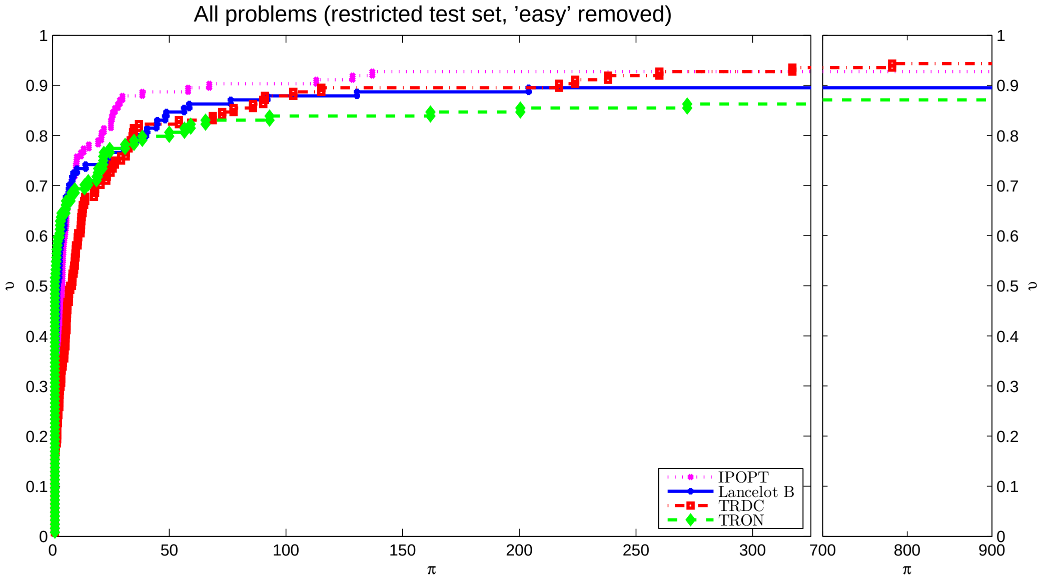

Figure 6: Performance profiles [12] for CPU time used by `IPOPT`, `Lancelot B`, `TRDC`, and `TRON`. All problems in the restricted test set, except the 'easy' ones.

References

[1] D. P. Bertesekas. Projected Newton methods for optimization problems with simple constraints. *SIAM J. Control Optim.*, 20:221–246, 1982.

[2] E. G. Birgin, J. M. Martínez, and M. Raydan. Nonmonotone spectral projected gradient methods on convex sets. *SIAM J. Optim.*, 10:1196–1211, 2000.

[3] R.H. Byrd, P. Lu, J. Nocedal, and C. Zhu. A limited memory algorithm for bound constrained optimization. *SIAM J. Sci. Comput.*, 16:1190–1208, 1995.

[4] F. H. Clarke. *Optimization and Nonsmooth Analysis*. John Wiley & Sons, New York, New York, 1983. Reissued by SIAM, Philadelphia, 1990.

[5] T. F. Coleman and Y. Li. An interior trust region approach for nonlinear minimization subject to bounds. *SIAM J. Optim.*, 6:418–445, 1996.

[6] 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 J. Numer. Anal.*, 25:433–460, 1988.

[7] A. R. Conn, N. I. M. Gould, and Ph. L. Toint. Correction to the paper on global convergence of a class of trust region algorithms for optimization with simple bounds. *SIAM J. Numer. Anal.*, 26:764–767, 1989.

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

[9] Y.-H. Dai. Fast algorithms for projection on an ellipsoid. *SIAM J. Optim.*, 16:986–1006, 2006.

[10] J. E. Dennis and L. N. Vicente. Trust-region interior-point algorithms for minimization problems with simple bounds. In H. Fisher, B. Riedmüller, and S. Schäffer, editors, *Applied Mathematics and Parallel Computing*, pages 97–107. Physica-Verlag, Springer-Verlag, Berlin, 1996. Festschrift for Klaus Ritter.

[11] J. E. Dennis Jr. and R. B. Schnabel. *Numerical Methods for Unconstrained Optimization and Nonlinear Equations*. Prentice–Hall, Englewood Cliffs, (republished by SIAM, Philadelphia, in 1996, as Classics in Applied Mathematics, 16), 1983.

[12] E. D. Dolan and J. J. Moré. Benchmarking optimization software with performance profiles. *Math. Program.*, 91:201–213, 2002.

[13] F. Facchinei, J. Júdice, and J. Soares. An active set Newton's algorithm for large-scale nonlinear programs with box constraints. *SIAM J. Optim.*, 8:158–186, 1998.

[14] A. Friedlander, J. M. Martínez, and S. A. Santos. A new trust region algorithm for bound constrained minimization. *Appl. Math. Optim.*, 30:235–266, 1994.

[15] N. I. M. Gould, D. Orban, and Ph. L. Toint. Contrained and unconstrainted test environement, revisited. `http://cuter.rl.ac.uk/cuter-www`.

[16] N. I. M. Gould, D. Orban, and Ph. L. Toint. Results from a numerical evaluation of LANCELOT B. Internal Report 2002-1, Numerical Analysis Group, Rutherford Appleton Laboratory, Chilton, England, 2002.

[17] N. I. M. Gould, D. Orban, and Ph. L. Toint. GALAHAD, a library of thread-safe Fortran 90 packages for large-scale nonlinear optimization. *ACM Trans. Math. Software*, 29:353–372, 2004.

[18] W. W. Hager and H. Zhang. A new active set algorithm for box constrained optimization. *SIAM J. Optim.*, 17:526–557, 2006.

[19] M. Heinkenschloss, M. Ulbrich, and S. Ulbrich. Superlinear and quadratic convergence of affine-scaling interior-point Newton methods for problems with simple bounds without strict complementarity assumption. *Math. Program.*, 86:615–635, 1999.

[20] H. A. Le Thi and T. Pham Dinh. Large scale molecular optimization from distance matrices by a D.C. optimization approach. *SIAM J. Optim.*, 14:77–114, 2003.

[21] H. A. Le Thi and T. Pham Dinh. The DC (difference of convex functions) programming and DCA revisited with DC models of real world nonconvex optimization problems. *Ann. Oper. Res.*, 133:23–46, 2005.

[22] M. Lescrenier. Convergence of trust region algorithms for optimization with bounds when strict complementarity does not hold. *SIAM J. Numer. Anal.*, 28:476–495, 1991.

[23] C.-J. Lin and J. J. Moré. TRON, a trust region Newton method for the solution of large bound-constrained optimization problems. http://www.mcs.anl.gov/∼more/tron/.

[24] C.-J. Lin and J. J. Moré. Newton's method for large bound-constrained optimization problems. *SIAM J. Optim.*, 9:1100–1127, 1999.

[25] J. L. Morales and J. Nocedal. Remark on "Algorithm 778. L-BFGS-B, Fortran subroutines for Large-Scale bound constrained optimization". *ACM Trans. Math. Software*, 38:7:1–7:4, 2011.

[26] J. Nocedal and S. J. Wright. *Numerical Optimization.* Springer-Verlag, Berlin, second edition, 2006.

[27] T. Pham Dinh and H. A. Le Thi. Convex analysis approach to D.C. programming: Theory, algorithms and applications. *Acta Math. Vietnam.*, 22:289–355, 1997.

[28] T. Pham Dinh and H. A. Le Thi. A D.C. optimization algorithm for solving the trust-region subproblem. *SIAM J. Optim.*, 8:476–505, 1998.

[29] R. T. Rockafellar. *Convex Analysis.* Princeton University Press, Princeton, 1970.

[30] R. T. Rockafellar and R. J.-B. Wets. *Variational Analysis.* Springer, Berlin, 1997, third printing in 2009.

[31] C. Sainvitu and Ph. L. Toint. A filter-trust-region method for simple-bound constrained optimization. *Optim. Methods Softw.*, 22:835–848, 2007.

[32] A. Schwartz and E. Polak. Family of projected descent methods for optimization problems with simple bounds. *J. Optim. Theory Appl.*, 92:1–31, 1997.

[33] A. I. F. Vaz and L. N. Vicente. A particle swarm pattern search method for bound constrained global optimization. *J. Global Optim.*, 39:197–219, 2007.

[34] A. Wächter and L. T. Biegler. On the implementation of an interior-point filter line-search algorithm for large-scale nonlinear programming. *Math. Program.*, 106:25–57, 2006.

[35] C. Zhu, R.H. Byrd, P. Lu, and J. Nocedal. Algorithm 778. L-BFGS-B, Fortran subroutines for Large-Scale bound constrained optimization. *ACM Trans. Math. Software*, 23:550–560, 1997.