Optimizing Radial Basis Functions by D.C. Programming and its use in Direct Search for Global Derivative-Free Optimization

Le Thi Hoai An* A. I. F. Vaz† L. N. Vicente‡

March 8, 2011

Abstract

In this paper we address the global optimization of functions subject to bound and linear constraints without using derivatives of the objective function. We investigate the use of derivative-free models based on radial basis functions (RBFs) in the search step of direct-search methods of directional type. We also study the application of algorithms based on difference of convex (d.c.) functions programming to solve the resulting subproblems which consist of the minimization of the RBF models subject to simple bounds on the variables. Extensive numerical results are reported with a test set of bound and linearly constrained problems.

Keywords: Global optimization, derivative-free optimization, direct-search methods, search step, radial basis functions, d.c. programming, DCA.

1 Introduction

We are interested in solving optimization problems of the form

$$\min_{x\in\mathbb{R}^n} \; f(x) \quad \text{s.t.} \quad x \in \Omega,$$

without using derivatives of the function f and when the feasible set Ω is a polyhedron. We will consider in more detail the case where Ω is solely defined by lower and upper bounds on the variables

$$\Omega = \{x \in \mathbb{R}^n : \ell \leq x \leq u\}. \tag{1}$$

*Laboratory of Theoretical and Applied Computer Science (LITA EA 3097), Paul Verlaine University, Metz, Ile du Saulcy, 57045 Metz, France (lethi@univ-metz.fr).

†Department of Systems and Production, University of Minho, Campus de Gualtar, 4710-057, Portugal (aivaz@dps.uminho.pt). Support for this author was provided by Algoritmi Research Center.

‡CMUC, Department of Mathematics, University of Coimbra, 3001-454 Coimbra, Portugal (lnv@mat.uc.pt). Support for this author was provided by FCT under research grants PTDC/MAT/64838/2006 and PTDC/MAT/098214/2008.

In (1) the inequalities $\ell \leq x \leq u$ are posed componentwise and $\ell \in (-\infty, \mathbb{R})^n$, $u \in (\mathbb{R}, +\infty)^n$, and $\ell < u$.

Our approach to address this type of problems is to incorporate the minimization of a radial basis functions (RBF) model in the search step of direct-search methods of directional type. This class of methods has been extensively studied in the literature (see the survey paper by Kolda, Lewis, and Torczon [18] or Chapter 7 of the book by Conn, Scheinberg, and Vicente [6]). Under appropriate assumptions they guarantee global convergence to stationary points. Their ability to find global minima depends on the incorporation of methods or heuristics for global optimization in their so-called search step. Two illustrative examples of such hybridizations are the approaches of Vaz and Vicente [33, 34], where a population-based heuristic was applied or, more recently, the approach of Griffin and Kolda [10], where DIRECT [15] was the chosen global optimization method.

On the other hand, models based on RBFs have been shown to be of interest for global optimization. In fact, derivative-free global optimization methods based on radial basis functions have been proposed by several authors (see the references [5, 11, 16] and the sequence of papers by Regis and Shoemaker [28, 29, 30, 31] which includes extensions to the constrained and parallel cases).

The overall direct-search algorithmic structure chosen for this paper is simple, but relatively efficient and robust as we will show by reporting extensive numerical experiments. In every iteration of direct search, we attempt to form an RBF model with as many points as possible and then to minimize it in the intersection of Ω with a trust region whose radius is proportional to the direct-search step size. We choose an ℓ_∞-shape trust region so that, in the case where Ω is given by (1), this intersection is still a box-constrained set. When Ω is defined by linear constraints not of the simple bound type, we apply an approximation procedure so that we still consider subproblems formed by the minimization of RBFs subject to box constraints. Our main conclusion is that a direct-search method with an RBF model minimization in the search step offers a good compromise between global optimization and computational effort (i.e., number of function evaluations) as long as relatively more than $n + 1$ points are used in the corresponding sampling process.

RBFs seem to offer a number of natural ways into which can be decomposed as a difference of two convex functions. In this paper, we will introduce two of such d.c. decompositions and adapt the d.c. algorithm (DCA) of [4] to both. The most efficient one is used to solve the subproblems which arise in the search step of the direct-search methods under consideration.

The structure of the paper is as follows. In Sections 2 and 3, we present some background material on radial basis functions and d.c. programming, respectively. The application of the DCA to the minimization of RBFs within simple bounds is studied in Section 4. The use of RBFs in the context of search steps of direct search is formally described in Section 5.

We then pause in Section 6 to describe our testing environment (test problems and forms of reporting the results). The numerical results are presented in two different sections. Numerical results for the different versions of the DCA algorithm when applied to minimize RBF models are shown in Section 7. We then provide extensive numerical

results involving the derivative-free solution of bound and linearly constrained problems in Section 8, comparing the use of RBF model minimization in direct search to other direct-search approaches. We conclude the paper in Section 9 with some final remarks.

2 Radial basis functions for optimization

In order to interpolate a function f whose values on a set $Y = \{y^1, \ldots, y^{n_p}\} \subset \mathbb{R}^n$ are known, one can use a radial basis functions (RBF) model of the form

$$m(x) \;=\; \sum_{i=1}^{n_p} \lambda_i \phi(\|x - y^i\|), \tag{2}$$

where $\phi(\|\cdot\|)$, with $\phi : \mathbb{R}_+ \to \mathbb{R}$, is a radial basis function and $\lambda_1, \ldots, \lambda_{n_p} \in \mathbb{R}$ are parameters to be determined.

For $m(x)$ to be twice continuously differentiable, the function $\phi(x)$ must be both twice continuously differentiable and have a derivative that vanishes at the origin. The cubic RBF, defined by $\phi(r) = r^3$, is among the most popular (twice continuously differentiable) radial basis functions, and has been frequently used for optimization purposes. Other popular RBFs, also twice continuously differentiable, are the Gaussian, the multiquadric, and the inverse multiquadric.

The term *radial basis* comes from the fact that $\phi(\|x\|)$ is constant on any sphere centered at the origin in $\mathbb{R}^n$. In many applications, it is desirable that the linear space spanned by the basis functions include constant or linear functions. Thus, it turns out to be useful to augment the radial basis function model in (2) by a low-order *polynomial tail* $\sum_{j=0}^{q} \gamma_j p_j(x)$, where p_j, $j = 0, \ldots, q$, are some basis functions for the polynomial and $\gamma_0, \ldots, \gamma_q \in \mathbb{R}$. The new model is now of the form

$$m(x) \;=\; \sum_{i=1}^{n_p} \lambda_i \phi(\|x - y^i\|) + \sum_{j=0}^{q} \gamma_j p_j(x).$$

Furthermore, the coefficients λ's are required to satisfy

$$\sum_{i=1}^{n_p} \lambda_i p_j(y^i) \;=\; 0, \quad j = 0, \ldots, q.$$

These, in conjunction with the interpolation conditions $m(y^i) = f(y^i)$, $i = 1, \ldots, n_p$, give the linear system

$$\begin{bmatrix} \Phi & P \\ P^\top & 0 \end{bmatrix} \begin{bmatrix} \lambda \\ \gamma \end{bmatrix} = \begin{bmatrix} f(Y) \\ 0 \end{bmatrix}, \tag{3}$$

where $\Phi_{ij} = \phi(\|y^i - y^j\|)$ for $i, j \in \{1, \ldots, n_p\}$, $P_{ij} = p_j(y^i)$ for $i \in \{1, \ldots, n_p\}$, $j \in \{0, \ldots, q\}$, and $f(Y)$ is the vector formed by the values $f(y^1), \ldots, f(y^{n_p})$.

The polynomial tails most frequently used in the context of RBFs are linear, and we will write $t(x) = c + g^\top x$ and

$$m(x) \;=\; \sum_{i=1}^{n_p} \lambda_i \phi(\|x - y^i\|) + t(x). \tag{4}$$

This model has $n_p + n + 1$ parameters, n_p for the radial basis terms and $n+1$ for the linear polynomial terms. However, when the number of points is $n+1$ (or less), the solution of the interpolation system gives rise to a linear polynomial, since all the parameters λ_i, $i = 1, \ldots, n_p$, are zero (see the second block equation in (3)). Consequently, the simplest nonlinear model $m(x)$ of the form (4) is based on $n+2$ interpolation points (and has $2n+3$ parameters).

We follow the approach by Oeuvray and Bierlaire [25, 26] and Wild, Regis, and Shoemaker [35, 36] for derivative-free optimization by using cubic radial basis functions and linear polynomial tails:

$$m(x) \;=\; \sum_{i=1}^{n_p} \lambda_i \|x - y^i\|^3 + t(x). \tag{5}$$

3 A brief review of d.c. programming

D.C. programming addresses the problem of minimizing a function f which is a difference of convex functions on the whole space $\mathbb{R}^n$ or on a convex set $C \subset \mathbb{R}^n$. Generally speaking, a d.c. program takes the form

$$\beta \;=\; \inf\{f(x) = g(x) - h(x) : x \in \mathbb{R}^n\}, \qquad (P_{dc})$$

where g and h are in the set $\Gamma_0(\mathbb{R}^n)$ of all lower semicontinuous proper convex functions in $\mathbb{R}^n$. Such a function f is called a d.c. function and $g-h$ the d.c. decomposition of f, while g and h are the d.c. components of f. The convex constraint $x \in C$ can be incorporated in the objective function of (P_{dc}) by using the indicator function on C denoted χ_C which is defined by $\chi_C(x) = 0$ if $x \in C$ and by $+\infty$ otherwise.

D.C. programming is one of the most relevant tools in nonsmooth nonconvex programming and global optimization. D.C. algorithms (DCA) were introduced by Pham Dinh Tao (see [7]) in their preliminary form in 1985 and have been extensively developed since 1994 by Le Thi Hoai An and Pham Dinh Tao, see [2, 3, 4] and the references therein. DCA have been successfully applied to many large-scale (smooth or nonsmooth) nonconvex programs in different fields of applied sciences for which they often give global solutions.

Recall that, for $\theta \in \Gamma_0(\mathbb{R}^n)$ and $\bar{x} \in dom\, \theta = \{x \in \mathbb{R}^n : \theta(\bar{x}) < +\infty\}$, the subdifferential $\partial\theta(\bar{x})$ of θ at $\bar{x}$ is defined as

$$\partial\theta(\bar{x}) \;=\; \{y \in \mathbb{R}^n : \theta(x) \geq \theta(\bar{x}) + (x - \bar{x})^\top y, \forall x \in \mathbb{R}^n\}.$$

A point x^* is critical or stationary for the minimization of $g - h$ when

$$\partial h(x^*) \cap \partial g(x^*) \;\neq\; \emptyset.$$

Let $g^*(y) := \sup\{x^\top y - g(x) : x \in \mathbb{R}^n\}$ be the conjugate function of g. Then, the following program is called the dual program of (P_{dc}):

$$\beta_D \;=\; \inf\{h^*(y) - g^*(y) : y \in \mathbb{R}^n\}. \qquad (D_{dc})$$

One can prove that $\beta = \beta_D$, (see, e.g., [4]) and there exists a perfect symmetry between the primal and dual d.c. programs: the dual of (D_{dc}) is exactly (P_{dc}).

The transportation of global solutions between (P_{dc}) and (D_{dc}) is expressed by:

$$[\cup_{y^* \in \mathcal{D}_{dc}} \partial g^*(y^*)] \;\subset\; \mathcal{P}_{dc}, \quad [\cup_{x^* \in \mathcal{P}_{dc}} \partial h(x^*)] \;\subset\; \mathcal{D}_{dc},$$

where $\mathcal{P}_{dc}$ and $\mathcal{D}_{dc}$ denote the solution sets of (P_{dc}) and (D_{dc}) respectively. Under certain conditions, this property also holds for the local solutions of (P_{dc}) and (D_{dc}), in the following sense. Let x^* be a local solution to (P_{dc}) and let $y^* \in \partial h(x^*)$. If g^* is differentiable at y^*, then y^* is a local solution to (D_{dc}). Similarly, let y^* be a local solution to (D_{dc}) and let $x^* \in \partial g^*(y^*)$. If h is differentiable at x^*, then x^* is a local solution to (P_{dc}).

Based on local optimality conditions and duality in DCA, the idea of DCA is quite simple: each iteration k of DCA approximates the concave part $-h$ by its affine majorization (that corresponds to taking $y^k \in \partial h(x^k)$) and minimizes the resulting convex function (that is equivalent to determining $x^{k+1} \in \partial g^*(y^k)$).

Generic DCA scheme
Initialization: Let $x^0 \in \mathbb{R}^n$ be a best guess, $0 \leftarrow k$.
Repeat
Calculate $y^k \in \partial h(x^k)$
Calculate $x^{k+1} \in \arg\min\{g(x) - h(x^k) - (x - x^k)^\top y^k : x \in \mathbb{R}^n\} \quad (P_k)$
$k + 1 \leftarrow k$
Until convergence of $\{x^k\}$.

Convergence properties of the DCA and its theoretical basis are described in [2, 3, 4]. However, for what comes next it is worthwhile to report the following properties:

- DCA is a descent method *without* line search, say the sequence $\{g(x^k) - h(x^k)\}$ is decreasing.
- If $g(x^{k+1}) - h(x^{k+1}) = g(x^k) - h(x^k)$, then x^k is a critical point of $g - h$. In this case, DCA terminates at the k-th iteration.
- If the optimal value β of problem (P_{dc}) is finite and the infinite sequence $\{x^k\}$ is bounded, then every limit point of this sequence is a critical point of $g - h$.
- DCA has a linear convergence for general d.c. programs.

Note that a d.c. function f has *infinitely many* d.c. decompositions and that the choice of a d.c. decomposition influences the speed of convergence, robustness, efficiency, and

globality (of computed solutions) of DCA. For a given d.c. program, the choice of the *optimal* d.c. decompositions is still open and depends strongly on the structure of the problem being considered. In practice, one chooses g and h such that the sequences $\{x^k\}$ and $\{y^k\}$ can be easily calculated.

4 Optimizing RBFs using d.c. algorithms

The optimization problem we are addressing in this section is

$$\min \;\; m(x) \;\; \text{s.t.} \;\; x \in \bar{\Omega}, \tag{6}$$

where $\bar{\Omega}$ is the model feasible region defined by upper and lower bounds on the variables, i.e., $\bar{\Omega} = \{x \in \mathbb{R}^n : \bar{\ell} \leq x \leq \bar{u}\}$. Since the feasible set $\bar{\Omega}$ may be different from (1), we are using here a different notation.

When ϕ is convex in $[0, +\infty)$, one possible d.c. decomposition of the RBF model (4) in $\bar{\Omega}$ is given by

$$m(x) \;=\; g_1(x) - h_1(x),$$

where

$$g_1(x) \;=\; \sum_{\lambda_i \geq 0} \lambda_i \phi(\|x - y^i\|) + t(x) + \chi_{\bar{\Omega}}(x), \qquad h_1(x) \;=\; \sum_{\lambda_i < 0} (-\lambda_i) \phi(\|x - y^i\|),$$

and $\chi_{\bar{\Omega}}(x)$ is the indicator function associated with $\bar{\Omega}$. The d.c. algorithm (DCA) corresponding to this decomposition is as follows.

Algorithm 4.1 (d.c. algorithm 1 (DCA1))

Initialization

Choose x_0.

For $k = 0, 1, 2, \ldots$

1. $y_k = \nabla h_1(x_k)$.
2. Compute x_{k+1} as the solution of

$$\min \;\; g_1(x) - \big(h_1(x_k) + (x - x_k)^\top y_k\big) \;\; \text{s.t.} \;\; x \in \bar{\Omega}. \tag{7}$$

Another possible d.c. decomposition for the RBF model (4) in $\bar{\Omega}$ is the following

$$m(x) \;=\; g_2(x) - h_2(x),$$

where

$$g_2(x) \;=\; \frac{\rho}{2}\|x\|^2 + t(x) + \chi_{\bar{\Omega}}(x), \qquad h_2(x) = \frac{\rho}{2}\|x\|^2 - (m(x) - t(x)),$$

$\chi_{\bar{\Omega}}(x)$ is, again, the indicator function associated with $\bar{\Omega}$, and

$$\rho \;=\; \max_{x\in\bar{\Omega}} \|\nabla^2(m(x) - t(x))\|.$$

Denoting by $P_{\bar{\Omega}}(\omega)$ the projection of ω onto the set $\bar{\Omega}$, the DCA becomes then the following.

Algorithm 4.2 (d.c. algorithm 2 (DCA2))

Initialization

Choose x_0. Compute ρ.

For $k = 0, 1, 2, \ldots$

1. $y_k = \nabla h_2(x_k)$.
2. $x_{k+1} = P_{\bar{\Omega}}(y_k/\rho)$.

An upper bound for ρ can be computed as follows when $\phi(r) = r^3$ and $\bar{\Omega}$ is defined as in (6):

$$\rho \;=\; 12 n_p \|\lambda\|_\infty \max\{\|\bar{\ell}\|, \|\bar{u}\|\}. \tag{8}$$

The derivation of this upper bound is based on the fact that when ϕ is twice continuously differentiable, one has

$$\nabla^2 m(x) \;=\; \sum_{i=1}^{n_p} \lambda_i \Theta(x - y^i),$$

with

$$\Theta(v) \;=\; \begin{cases} \frac{\phi'(\|v\|)}{\|v\|} I + \left(\phi''(v) - \frac{\phi'(\|v\|)}{\|v\|}\right) \frac{v}{\|v\|}\frac{v^\top}{\|v\|} & \text{if } v \neq 0, \\ \phi''(0) I & \text{if } v = 0. \end{cases}$$

A numerical study of these two algorithms is reported in Section 7.

5 Using RBF modeling in direct search

Now we investigate the use of radial basis function models in direct-search methods for global derivative-free optimization. Our approach consists of forming and minimizing an RBF model in the search step of direct-search methods of the directional type. The iterations of such methods can be divided into two main steps (a search step and a poll step). For the purposes of global convergence to stationary points, the search step is optional and free of any rules except that it must terminate finitely and yield a point in the underlying integer lattice.

The algorithmic description below applies to the case where Ω is solely defined by bound constraints (1). In this situation, the poll step makes use of a constant positive spanning set that includes the coordinate vectors and their negatives, which conform to the feasible set. The linearly constrained case is discussed afterwards.

Algorithm 5.1 (Direct-search method (RBFs in search step))

Initialization

Choose x_0, $\alpha_0 > 0$, and $0 < \sigma_{min} \leq \sigma_{max}$. Let $D = \{e_1, \ldots, e_n, -e_1, \ldots, -e_n, e, -e\}$ (where e_i denotes the i-th column of the identity and e the vector of ones, of dimensions n).

For $k = 0, 1, 2, \ldots$

1. **Search step:** Form an RBF model $m(x)$ of the type (4) based on a subset of previously evaluated points and minimize a subproblem of the form (6) where $\bar{\Omega} = \Omega \cap B(x_k; \Delta_k)$, Ω is given by (1), $B(x_k; \Delta_k) = \{x \in \mathbb{R}^n : \|x - x_k\|_\infty \leq \Delta_k\}$, and $\Delta_k = \sigma_k \alpha_k$, with $\sigma_k \in [\sigma_{min}, \sigma_{max}]$.
   If the solution $\bar{x}_k$ of the subproblem satisfies $f(\bar{x}_k) < f(x_k)$, then set $x_{k+1} = \bar{x}_k$, declare the iteration and the search step successful, and skip the poll step.
2. **Poll step:** Optionally order the poll set $P_k = \{x_k + \alpha_k d : d \in D\}$. The points in P_k can be ordered by increasing values of the RBF model. If a poll point $x_k + \alpha_k d_k$ is found such that $f(x_k + \alpha_k d_k) < f(x_k)$ then stop polling, set $x_{k+1} = x_k + \alpha_k d_k$, and declare the iteration and the poll step successful. Otherwise declare the iteration (and the poll step) unsuccessful and set $x_{k+1} = x_k$.
3. **Step size update:** If the iteration was successful then maintain the step size parameter ($\alpha_{k+1} = \alpha_k$) or double it ($\alpha_{k+1} = 2\alpha_k$) after two consecutive poll successes along the same direction. If the iteration was unsuccessful, halve the step size parameter ($\alpha_{k+1} = \alpha_k/2$).

The subproblem considered in the search step consists of the minimization of an RBF model on the intersection of Ω with an ℓ_∞-shape trust region. When Ω is defined by bounds, as in (1), this amounts to a box-constrained domain and the DCAs of the previous section can be easily applied.

If Ω contains linear constraints which are not simple bounds, then we simplify the subproblem by temporarily removing those constraints. The point to be evaluated in the search step is then defined by $x_k + \min\{1, \tau_k\}(\bar{x}_k - x_k)$, where τ_k is the largest positive scalar such that $x_k + \min\{1, \tau_k\}(\bar{x}_k - x_k) \in \Omega$. This procedure is the same as the one used in [34] to address infeasible points in the search step.

To obtain a feasible initial guess and a set of poll directions which conforms to the feasible set, when Ω contains linear constraints which are not simple bounds, we also use the same strategies as in [34].

A numerical study of Algorithm 5.1 is reported in Section 8 for both bound and linearly constrained problems.

6 Test problems and profiles

The numerical results reported on Sections 7 and 8 were conducted in MATLAB R14 (7.0.1)[1] and were run in a Pentium Centrino (2.0GHz and 3Gb of RAM).

6.1 Test problems

The test set used in the numerical results consists of 107 bound constrained problems and 92 linearly constrained problems coded in the AMPL format. The choice of AMPL [9] made the process of coding a large number of problems faster. Since the algorithms were coded in MATLAB, an AMPL-MATLAB interface was used, as described in [33, 34].

The bound constrained problems are the ones reported in [33]. These problems are global optimization test problems collected from the literature [1, 12, 13, 17, 19, 20, 23, 27] and coded in AMPL. All of these problems have lower and upper bounds on the variables. The linearly constrained problems correspond to those already used for testing in [34], and also coded in AMPL. They were gathered from the internet[2] and from the papers [14, 21, 22, 32]. The problems descriptions and their sources are available at `http://www.norg.uminho.pt/aivaz/pswarm`.

The problems used are listed in Table 1 for the bound constrained case and in Table 2 for the linearly constrained case. Tables include the problem names and corresponding dimensions (n).

6.2 Performance and data profiles

For a better visualization, brevity, and clarity of the numerical results, we are providing performance profiles obtained by using the procedure described in [33] (a modification of the performance profiles from [8]). The major advantage of the performance profiles is that they can be presented in one figure, by plotting, for the different solvers, a cumulative distribution function $\upsilon(\tau)$ 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 the value obtained by solver s on test problem p. Then, define $\upsilon_s(\tau) = \frac{1}{N_p}\text{size}\{p \in \mathcal{P} : r_{p,s} \leq \tau\}$, where N_p is the number of test problems. The value of $\upsilon_s(1)$ is the probability that the solver s will win over the remaining ones (meaning that it will yield a value lower than the values of the remaining ones). If we are only interested in determining which solver is the best (in the sense that

[1] `www.mathworks.com`.

[2] `www.sor.princeton.edu/~rvdb/ampl/nlmodels/index.html`, `http://cuter.rl.ac.uk/cuter-www`, `http://www.gamsworld.org/global/globallib.htm`, `www.mat.univie.ac.at/~neum/glopt/coconut/Benchmark/Benchmark.html`.

Name	n	Name	n	Name	n	Name	n	Name	n	Name	n
ack	10	fx_10	10	ir5	2	ml_10	10	rg_10	10	sz	1
ap	2	fx_5	5	kl	4	ml_5	5	rg_2	2	szzs	1
bf1	2	gp	2	ks	1	mr	3	s10	4	wf	4
bf2	2	grp	3	lm1	3	mrp	2	s5	4	xor	9
bhs	2	gw	10	lm2_10	10	ms1	20	s7	4	zkv_10	10
bl	2	h3	3	lm2_5	5	ms2	20	sal_10	10	zkv_2	2
bp	2	h6	6	lms1a	2	nf2	4	sal_5	5	zkv_20	20
cb3	2	hm	2	lms1b	2	nf3_10	10	sbt	2	zkv_5	5
cb6	2	hm1	1	lms2	3	nf3_15	15	sf1	2	zlk1	1
cm2	2	hm2	1	lms3	4	nf3_20	20	sf2	2	zlk2a	1
cm4	4	hm3	1	lms5	6	nf3_25	25	shv1	1	zlk2b	1
da	2	hm4	2	lv8	3	nf3_30	30	shv2	2	zlk3a	1
em_10	10	hm5	3	mc	2	osp_10	10	sin_10	10	zlk3b	1
em_5	5	hsk	2	mcp	4	osp_20	20	sin_20	20	zlk3c	1
ep	2	hv	3	mgp	2	prd	2	stg	1	zlk4	2
exp	10	ir1	3	mgw_10	10	ptm	9	st_17	17	zlk5	3
fls	2	ir2	2	mgw_2	2	pwq	4	st_9	9	zzs	1
fr	2	ir4	30	mgw_20	20	rb	10	swf	10		

Table 1: Bound constrained problems used in the numerical results.

Name	n	Name	n	Name	n	Name	n
antenna2	24	ex2_1_6	10	hs21mod	7	s253	3
avgasa	6	ex2_1_7	20	hs268	5	s268	5
avgasb	6	expfita	5	hs35mod	2	s277	4
biggsc4	4	expfitb	5	hs44new	4	s278	6
bunnag1	3	expfitc	5	hubfit	2	s279	8
bunnag2	4	fir_linear	11	Ji1	3	s280	10
bunnag3	5	g01	13	Ji2	3	s331	2
bunnag4	6	genocop07	6	Ji3	2	s340	3
bunnag5	6	genocop09	3	ksip	20	s354	4
bunnag6	10	genocop10	4	lowpass	31	s359	5
bunnag7	10	genocop11	6	lsqfit	2	s392	30
bunnag8	20	goffin	51	makela4	21	simpllpa	2
bunnag9	20	gtm	59	Michalewicz1	2	simpllpb	2
bunnag10	20	hatfldh	4	nuffield_continuum	2	sipow1	2
bunnag11	20	hs021	2	oet1	3	sipow1m	2
bunnag12	20	hs024	2	oet3	4	sipow2	2
bunnag13	20	hs035	3	pentagon	6	sipow2m	2
ex2_1_1	5	hs036	3	pt	2	sipow3	4
ex2_1_10	20	hs037	3	s224	2	sipow4	4
ex2_1_2	6	hs044	4	s231	2	stancmin	3
ex2_1_3	13	hs076	4	s232	2	tfi2	3
ex2_1_4	6	hs086	5	s250	3	weapons	65
ex2_1_5	10	hs118	15	s251	3	zecevic2	2

Table 2: Linearly constrained problems used in the numerical results.

wins the most), we compare the values of $\upsilon_s(1)$ for all the solvers. At the other end, solvers with the largest probabilities $\upsilon_s(\tau)$ for large values of τ are the most robust ones (meaning that are the ones that solved the largest number of problems).

On the other hand, one possible way to assess solvers performance given a certain budget of objective function evaluations is given by the so-called data profiles [24]. For each solver, a data profile consists of a plot of the percentage of problems that are solved for a given budget of function evaluations. Let $\eta_{p,s}$ be the number of function evaluations required for solver $s \in \mathcal{S}$ to solve problem $p \in \mathcal{P}$ (up to a certain accuracy). The data profile cumulative function is then defined by

$$d_s(\sigma) \;=\; \frac{1}{N_p}\text{size}\left\{p \in \mathcal{P} : \eta_{p,s} \leq \sigma\right\}. \tag{9}$$

This definition for $d_s(\sigma)$ is independent of the number of variables in the problem. As the number of function evaluations (required to determine some solution within certain accuracy) is expected to increase with the increase in the number of variables, one can change the definition of $d_s(\sigma)$ to

$$d_s(\sigma) \;=\; \frac{1}{N_p}\text{size}\left\{p \in \mathcal{P} : \frac{\eta_{p,s}}{n^p+1} \leq \sigma\right\} \tag{10}$$

where n^p is the number of variables for problem $p \in \mathcal{P}$.

A critical issue related to data profiles is how to consider a problem as being solved. The authors in [24] suggested that a problem is solved (up to some level ε of accuracy) when

$$f(x_0) - f(x) \;\geq\; (1-\varepsilon)(f(x_0) - \bar{f}), \tag{11}$$

where x_0 is the initial guess and $\bar{f}$ is the best obtained objective function value among all solvers when a given number of function evaluations is imposed.

7 Testing DCA for the minimization of the RBF models

Since the implementation of the DCA algorithms described in Section 4 was done in MATLAB, we decided to compare them to the `fmincon` MATLAB solver. For this purpose, and given that we are mainly interested in minimizing RBFs within simple bounds (i.e., problems of the form (6), in particular when $m(x)$ is given by (5)), we have used the part of the test set described at Subsection 6.1 formed only by bound constrained problems (thus setting $\bar{\Omega}$ in (6) as in the original problems (1)).

Four algorithms were used in this comparison. The first, DCA1, is an implementation of Algorithm 4.1 (using `fmincon` to solve the subproblems (7)). The `fmincon` starting point was the final point obtained in the previous iteration (except in the first iteration where we have used a random point). The `fmincon` stopping tolerances were set to 10^{-1}.

The stopping criterion of DCA1 was satisfied when the absolute error of two consecutive iterations did not exceed 10^{-5} or the number of iterations exceeded 30000.

The two next instances subject to testing are based on Algorithm 4.2. In both cases, the stopping criterion was satisfied when the absolute error of two consecutive iterations did not exceed 10^{-5} or the number of iterations exceeded 30000. In the first version (DCA2a), the value of ρ is kept constant as in (8). In the second version (DCA2b), we started with an initial value for ρ given by (8) multiplied by 5×10^{-6}, and increased it by a factor of two each time a new point did not lead to a simple decrease ($m(x_{k+1}) < m(x_k)$) in the objective function value (i.e., the RBF model in question), until the upper bound in (8) was reached.

The fourth algorithm used to solve problem (6) was `fmincon` itself, setting the stopping tolerances to 10^{-5}. Only first-order derivatives were provided for all the four algorithms.

Numerical results were considered when the RBF model is built by using $n_p = 3n + 1$, $5n + 1$, 10, 20, 50, and 100 points. These points are randomly generated in the feasible region $\bar{\Omega}$ of (6) given by (1) until the linear system (3) produces finite values for λ and γ (and no further control on λ and γ was made).

Since the initial guess for all the algorithms is randomly generated, we have performed 10 and 20 runs of the algorithms for all the bound constrained problems in the test set. Since no major differences were observed, we are reporting the numerical results based on 10 runs. Average, maximum, and minimum metrics were calculated, but, for the sake of brevity, we report only the average ones (the maximum and minimum ones do not change any of the conclusions below).

Illustrative performance profiles are reported in Figures 1 to 2 based on the average CPU time used by each algorithm. From these plots, we can observe that DCA2b and `fmincon` exhibit a similar performance with respect to the average CPU time, while DCA1 and DCA2a are worse, taking, in average, more time to solve the problems.

In Figures 3 and 4 we report the average obtained objective function values. From these plots we can conclude that DCA1 and DC2b are the ones obtaining on average the best objective function values, while DCA2a and `fmincon` are slightly worse.

To finish this assessment of performance, we report the percentage of problems solved within increasing budgets of objective function evaluations. Such data profiles are given in Figures 5-8. We removed DCA1 from this comparison, as it does not use the model $m(x)$ explicitly and we are measuring complete RBF model evaluations. Since data profiles rely on the accuracy parameter ε, we present two plots in each figure, for $\varepsilon = 10^{-1}$ and $\varepsilon = 10^{-5}$. Recall from (11) that we are requiring more accuracy for $\varepsilon = 10^{-5}$ than for $\varepsilon = 10^{-1}$. From the plots, we can observe that DCA2b is the algorithm that solves in average more problems for a given budget. These profiles were calculated based on equation (10), but similar results were obtained with equation (9). Again we are reporting data profiles based on 10 runs for each solver. Note that the value of $\bar{f}$ was determined by first running all the solvers with a maximum number of 2000 function evaluations and then taking the minimum value obtained among them.

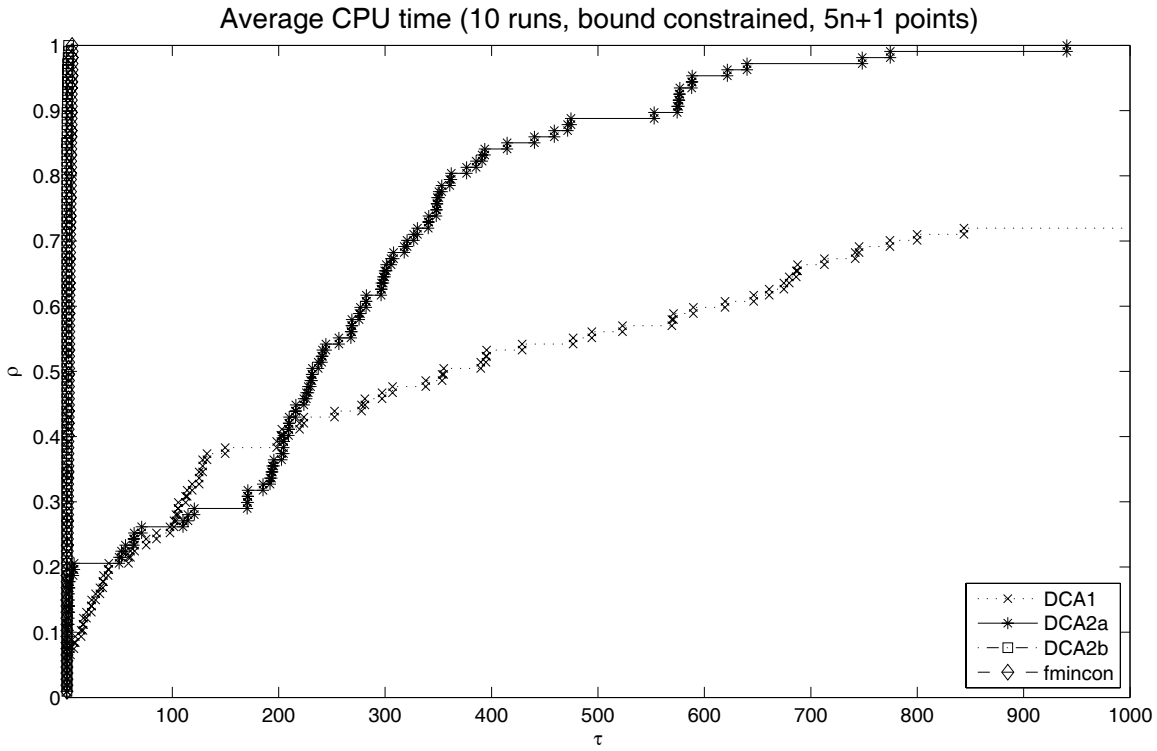

Figure 1: Performance profiles for average CPU time used by DCA1, DCA2a, DCA2b, and `fmincon`.

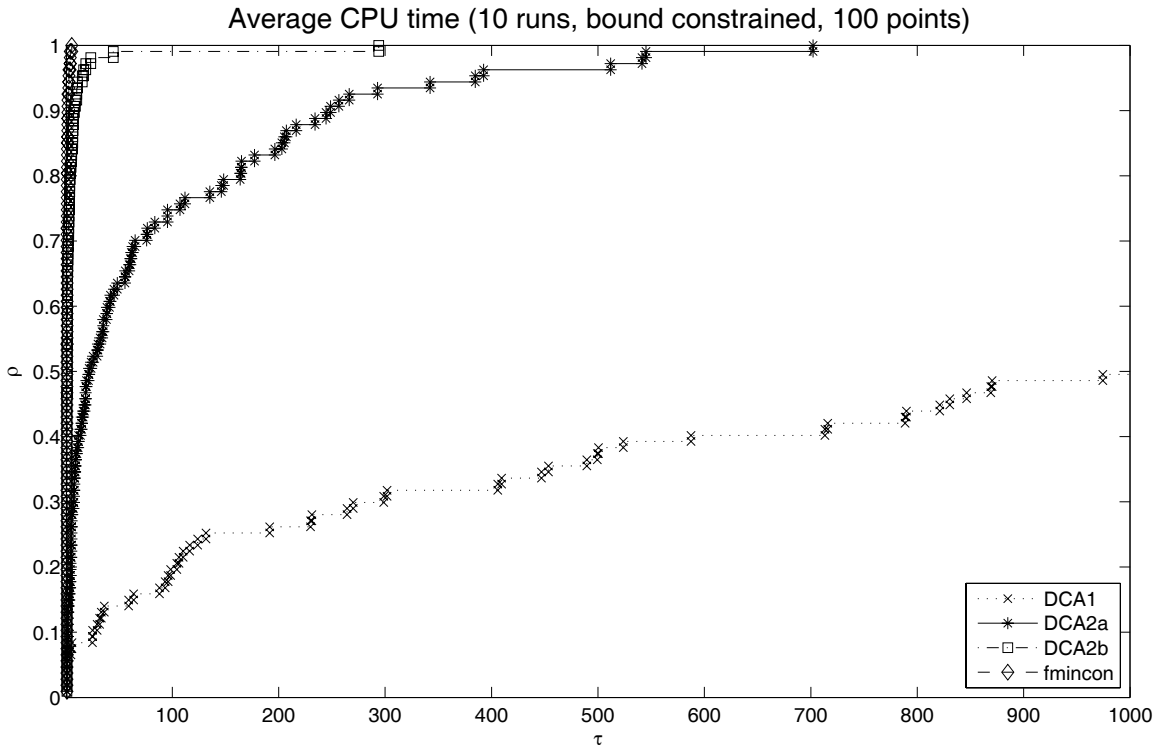

Figure 2: Performance profiles for average CPU time used by DCA1, DCA2a, DCA2b, and `fmincon`.

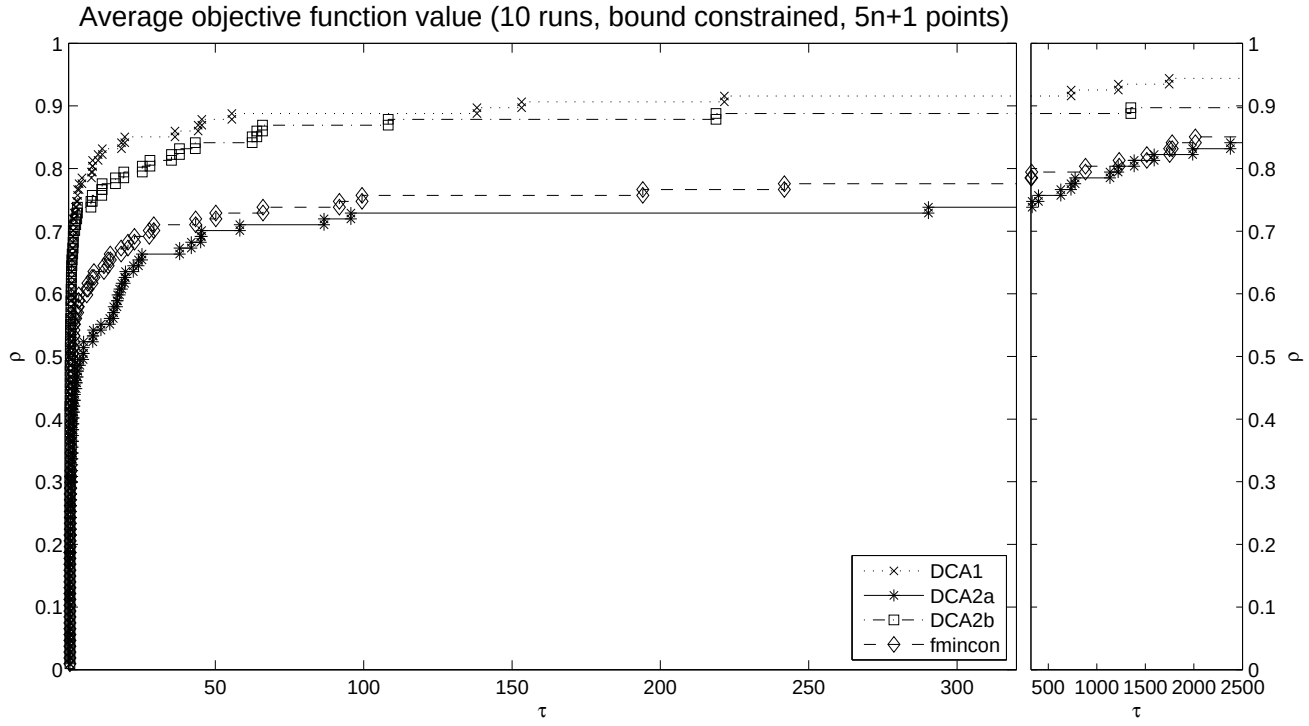

Figure 3: Performance profiles for average objective function value obtained by DCA1, DCA2a, DCA2b, and `fmincon`.

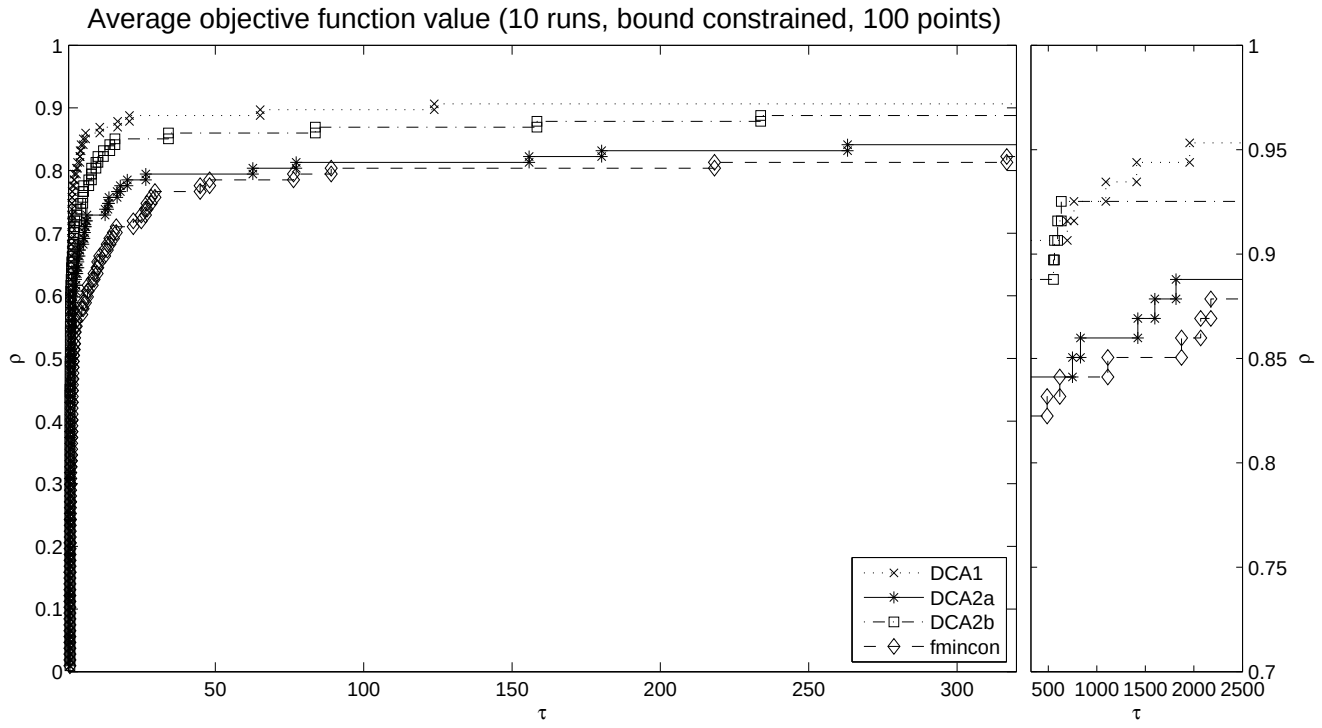

Figure 4: Performance profiles for average objective function value obtained by DCA1, DCA2a, DCA2b, and `fmincon`.

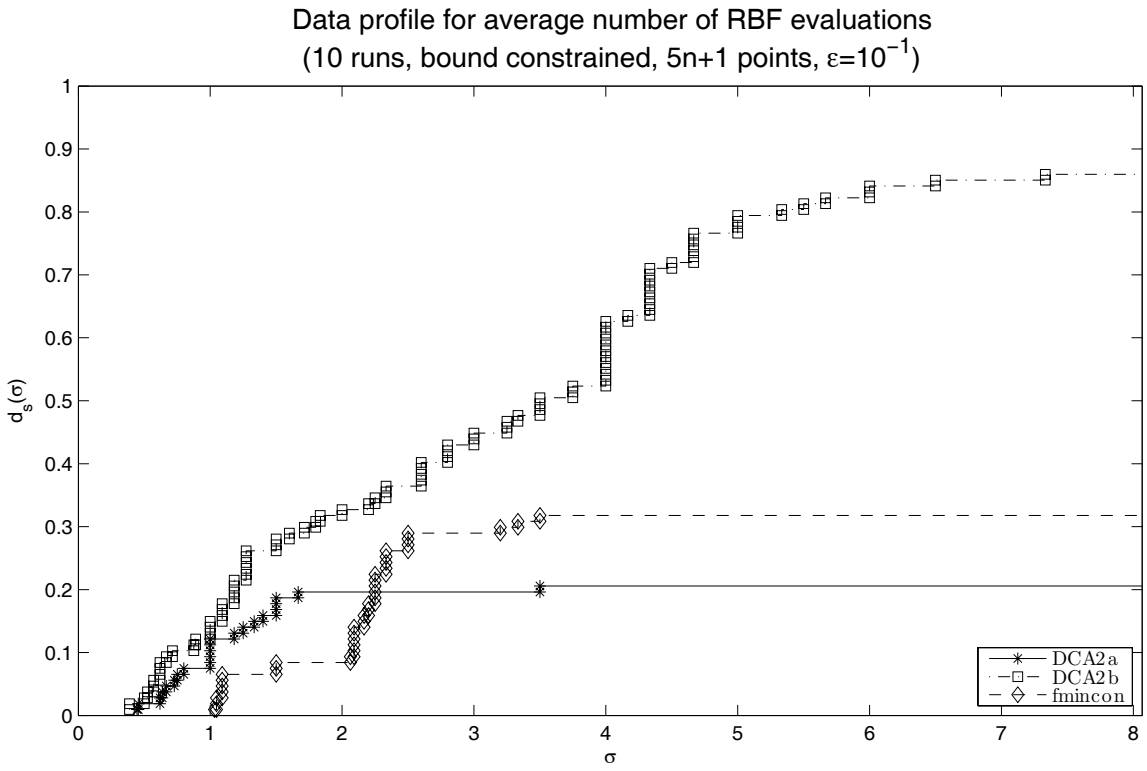

Figure 5: Data profile for average number of RBF evaluations obtained by DCA2a, DCA2b, and `fmincon`.

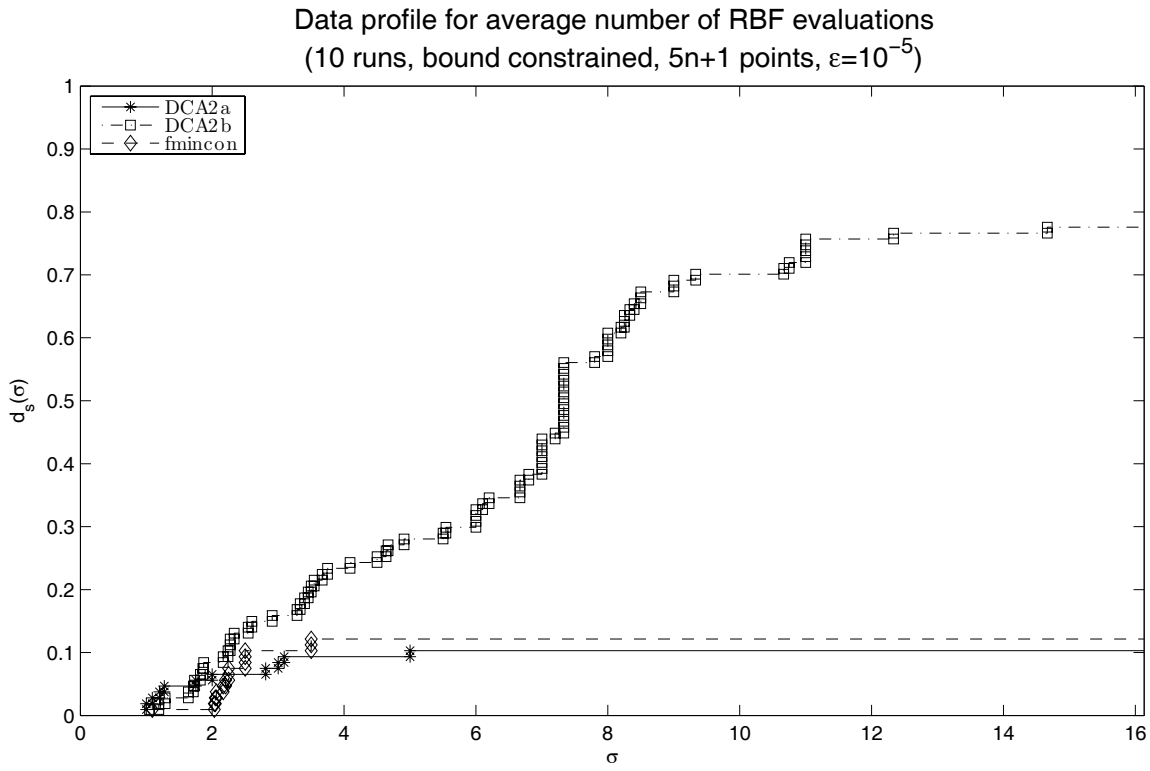

Figure 6: Data profile for average number of RBF evaluations obtained by DCA2a, DCA2b, and `fmincon`.

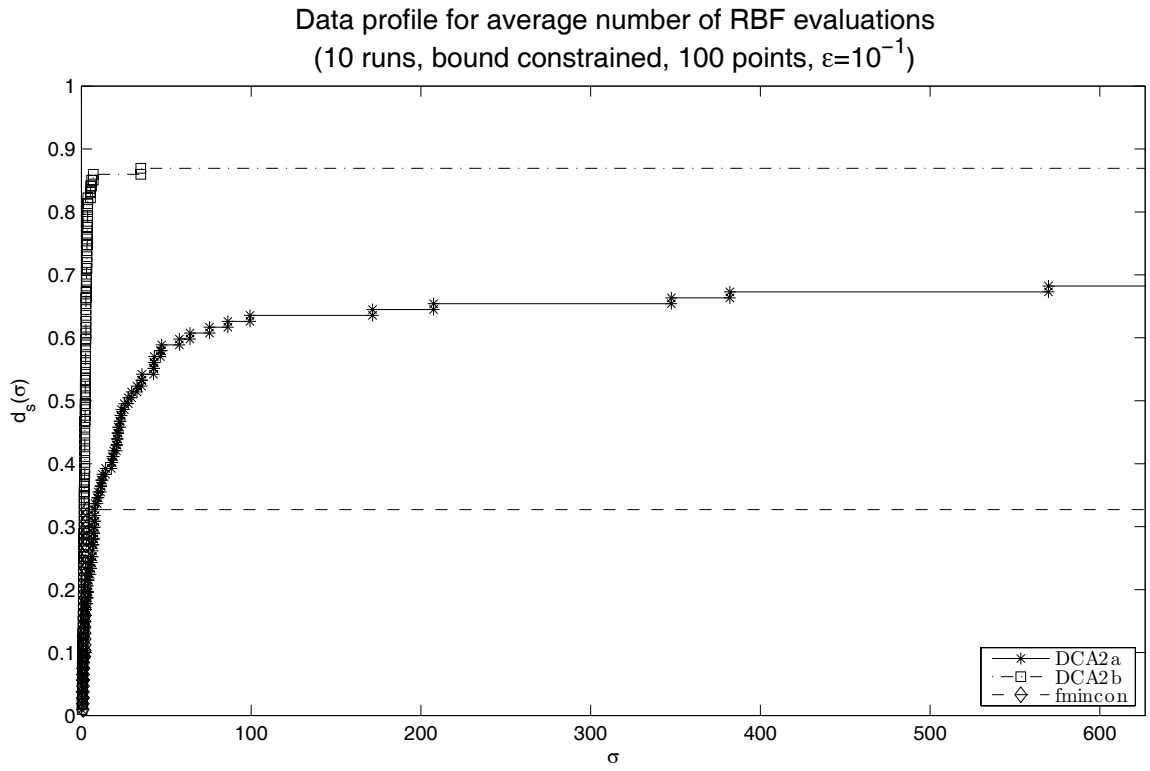

Figure 7: Data profile for average number of RBF evaluations obtained by DCA2a, DCA2b, and `fmincon`.

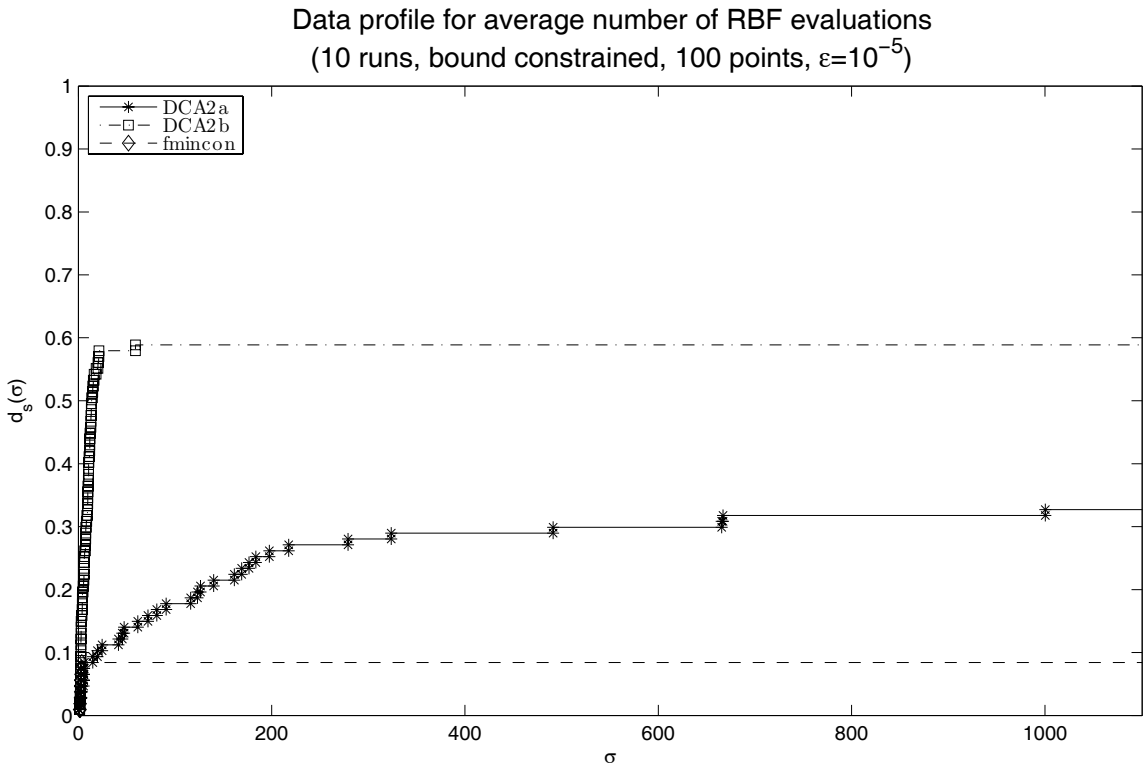

Figure 8: Data profile for average number of RBF evaluations obtained by DCA2a, DCA2b, and `fmincon`.

8 Testing the use of RBF model minimization in direct search

We chose to implement Algorithm 5.1 (using RBF modeling in the search step) within the PSwarm solver, taking advantage of the availability of an existing direct-search implementation. For a comparison of the PSwarm solver to other available solvers, we point readers to [33, 34]. In this paper we are only interested in accessing the advantage of using an RBF model in the search step of a direct-search method.

Taking advantage of the previous implementations of the algorithm described in [33, 34], we made use of a cache for the true function evaluations. This cache can be used by both the search and poll steps when a function evaluation is requested. The cache proved not to be useful in the particle swarm search step, as the number of hits in the cache was very low. But, for the direct-search version proposed in this paper, the cache implementation is extremely important, since the RBF model building relies on previous evaluated objective function values. The maximum number of points in cache considered for the RBF model building was $50(n+1)$.

8.1 Building the RBF models

The RBF models are built in the search step by setting $n_{min} = n+2$ and $n_{max} = 5n+1$ as the minimum and maximum number of points considered in the corresponding sample sets. The search step is skipped if the number of points previously evaluated is less than n_{min}. All previously evaluated points are used to build the model if its number is lower than n_{max}. However, if there are more previously evaluated points in the cache than n_{max}, then 80% of the points for model building are selected as the ones nearest to the current iterate. The last 20% are chosen as the ones further away from the current iterate. This strategy is used in order to improve the quality of the geometry of the sample sets in the spirit of the implementation of some interpolation-based trust-region methods [6, Chapter 11]. Other percentages have been tested, but these ones have shown to be most appropriate among those tried.

In the search step of Algorithm 5.1, we set σ_k to 1 if the previous search step was unsuccessful, or to 2 otherwise.

8.2 Numerical results for the use of RBF model minimization in direct search

We provide aggregated results for all the test problems, results for the class of bound constrained problems, and results for the class of linearly constrained problems. Showing the results in this way allows us to suggest different algorithmic options for each class of problems.

We are reporting numerical results for the following versions of Algorithm 5.1: RBF search step with DCA as the algorithm used to solve subproblem (6) (RBF-DCA); RBF

search step with `fmincon` as the algorithm used to solve subproblem (6) (RBF-fmincon); RBF search step with DCA as the algorithm to solve subproblem (6) and the directions used in the poll step sorted by the RBF model value at the poll points (RBF-DCA Sort); RBF search step with `fmincon` as the algorithm used to solve subproblem (6) and the directions used in the poll step sorted by the RBF model value at the poll points (RBF-fmincon Sort). We compared these four instances against a version of Algorithm 5.1 with an empty search step (Pattern) and the PSwarm implementation (particle swarm search step). In all cases, the stopping criterion consisted of reaching a maximum budget of 1000 function evaluations or driving the step size parameter α_k below 10^{-5}.

The algorithm used in the RBF-DCA versions is the DCA2b described in Section 4 for cubic RBFs. A limit of 3000 iterations for the DCA2 algorithm was set, as an approximation with high accuracy to the solution of subproblem (6) is not required. Even when the DCA algorithm is unable to converge within the requested accuracy, the last iterate that it produces is still used to check for progress in the objective function in the search step.

Figure 9 presents a comparison between the DCA algorithm and `fmincon` when using a RBF model in the search step. Additionally we include the case where the search directions of the poll step are sorted accordingly to the RBF model. For this comparison, we have set $t_{p,s} = \frac{b}{a}$, where a is the number of models leading to an improvement in the objective function and b is the number of models built, corresponding to the inverse of the percentage of success in using the RBF model. When none of the model building leads to a success we have $t_{p,s} = +\infty$. Observing the profiles, we can see that in nearly 30% of the problems all the versions where able to obtain the best $t_{p,s}$ value (τ around 1). Looking at large values of τ, we observe that in 90% of the problems the model has a nonzero success rate. In this comparison among success rates, the specific profiles for bound and linearly constrained problems are omitted, since they provide similar results.

Figures 10–12 depict profiles for the quality of the final objective function value obtained. We can easily verify that all the tested solvers present no major advantage over the remaining ones. As expected, PSwarm showed ability to overcome some non-convexity of the objective function. However, we recall that PSwarm is a population based algorithm that often uses all the objective function evaluations budget.

The data profiles for function evaluations are reported in Figures 13–18 showing an advantage of the DCA2b version over the others for solving problems within given budgets and independently of the accuracy considered. In this case, a maximum number of 500 function evaluations was imposed to compute the $\bar{f}$ value in (11).

9 Conclusions

In this paper we proposed the use of radial function basis (RBF) models to improve the efficiency of a direct-search type method for the global optimization of functions subject to bound and linear constraints. The RBF models are known to model well multimodal functions and proved here to be useful in the context of black-box optimization of functions expensive to evaluate.

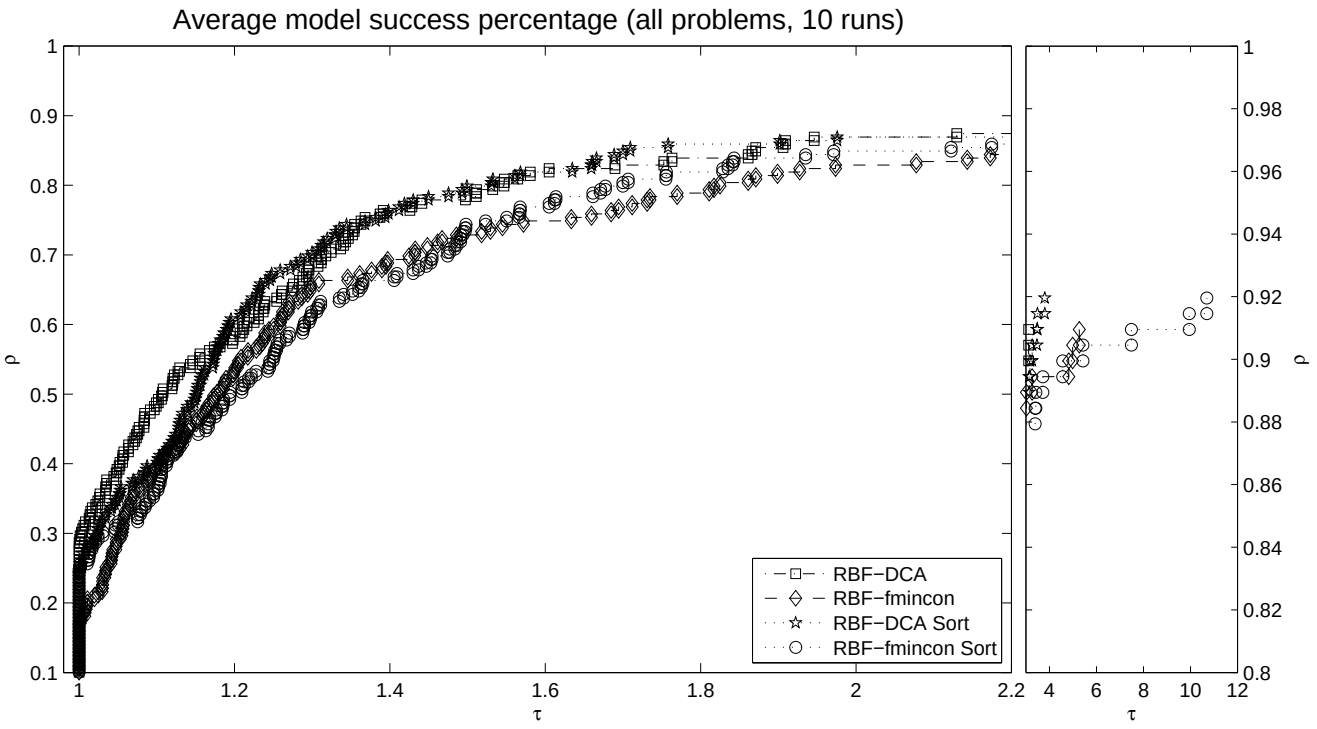

Figure 9: Comparison among RBF-DCA, RBF-fmincon, RBF-DCA Sort, and RBF-fmincon Sort, for all test problems (performance profiles for percentage of built RBF models leading to successful iterations).

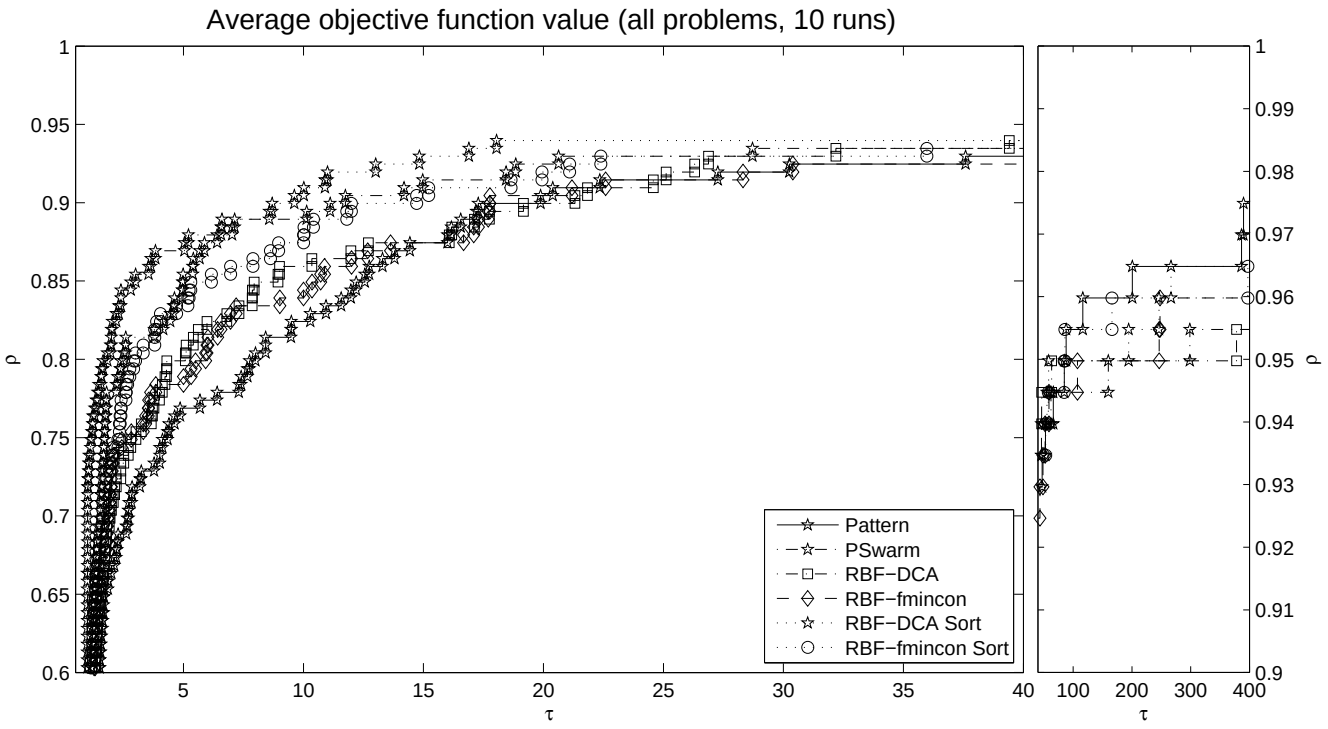

Figure 10: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for all problems (performance profiles for average objective function value).

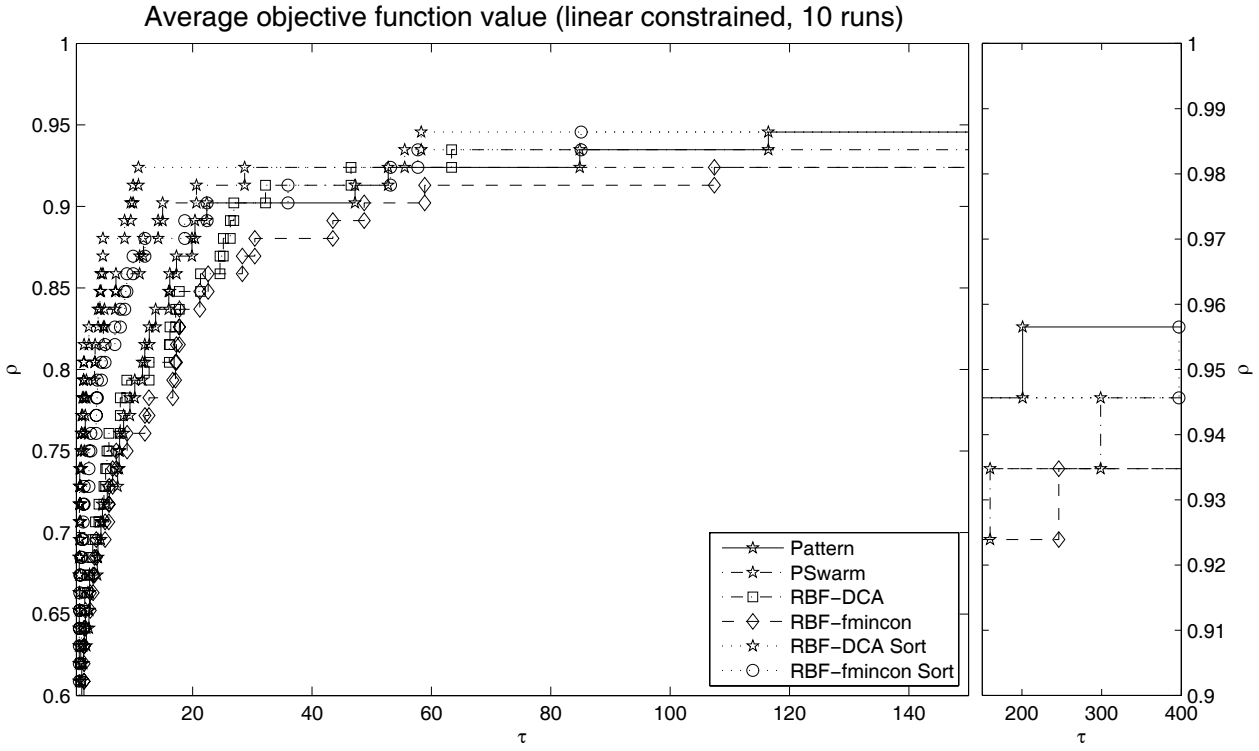

Figure 11: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for linearly constrained problems (performance profiles for average objective function value).

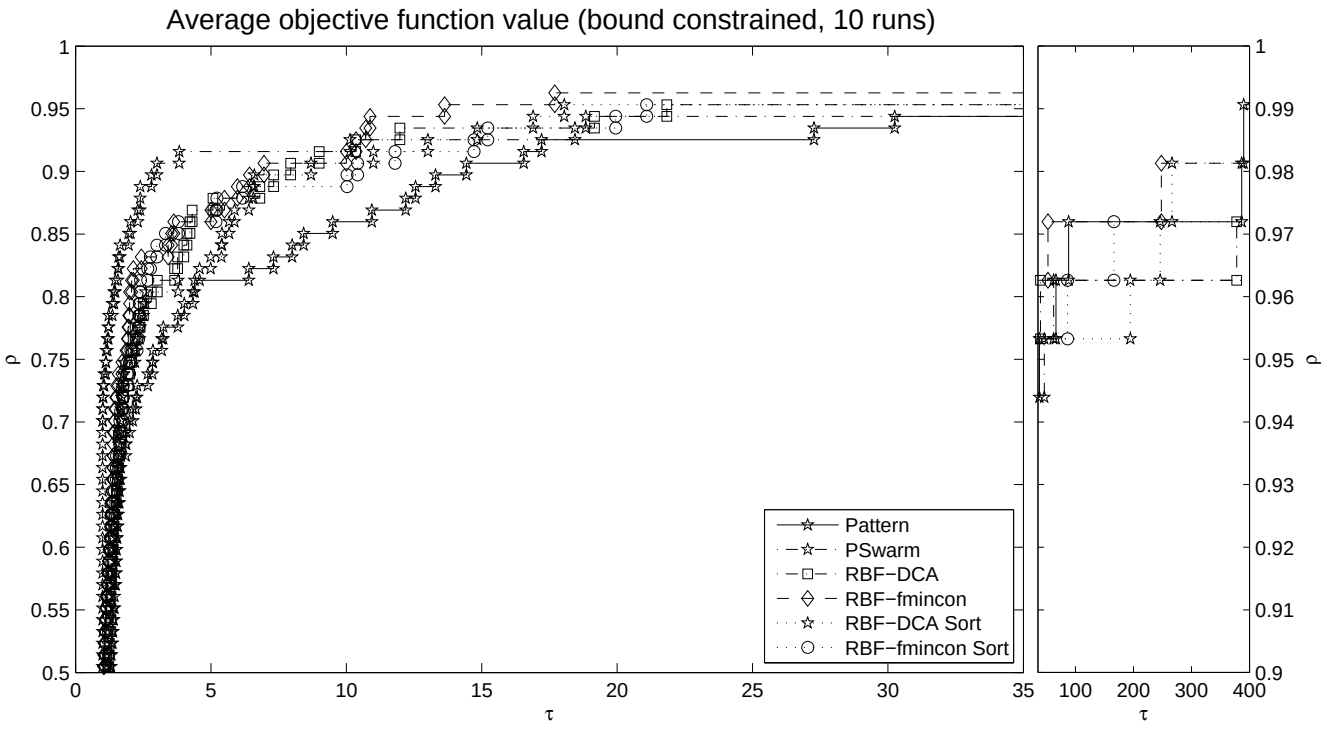

Figure 12: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for bound constrained problems (performance profiles for average objective function value).

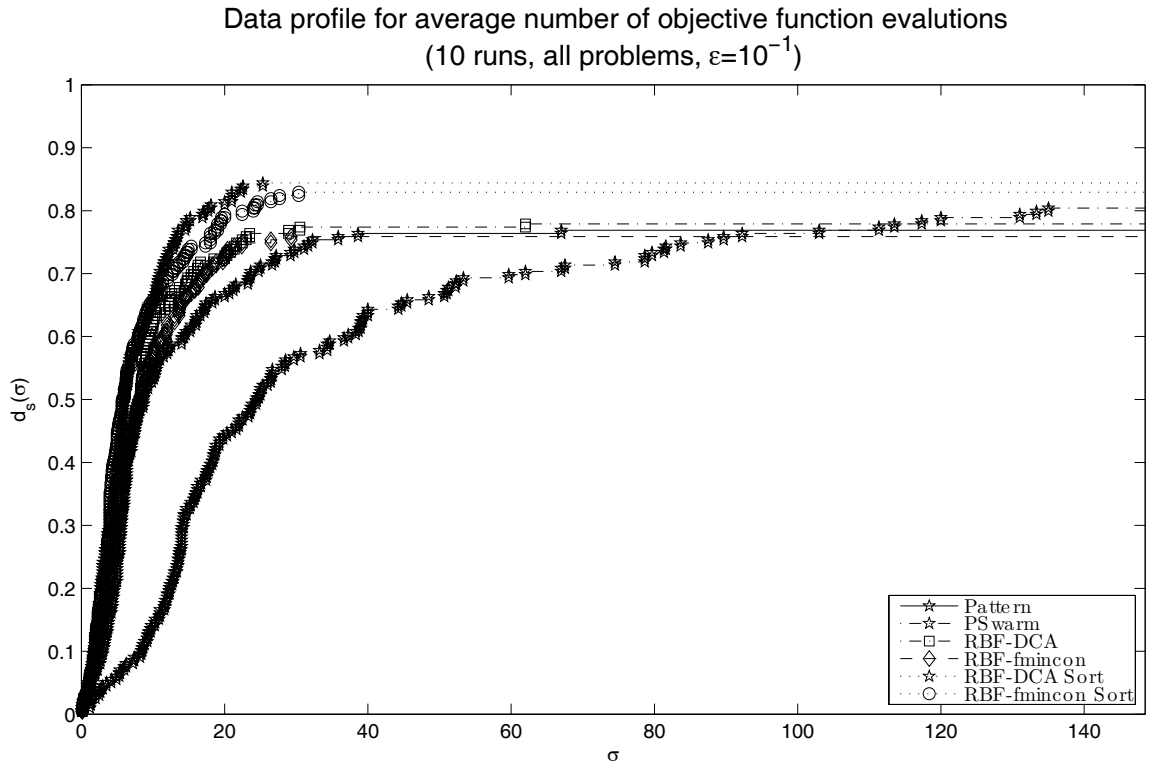

Figure 13: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for all problems (data profiles for average number of objective function evaluations).

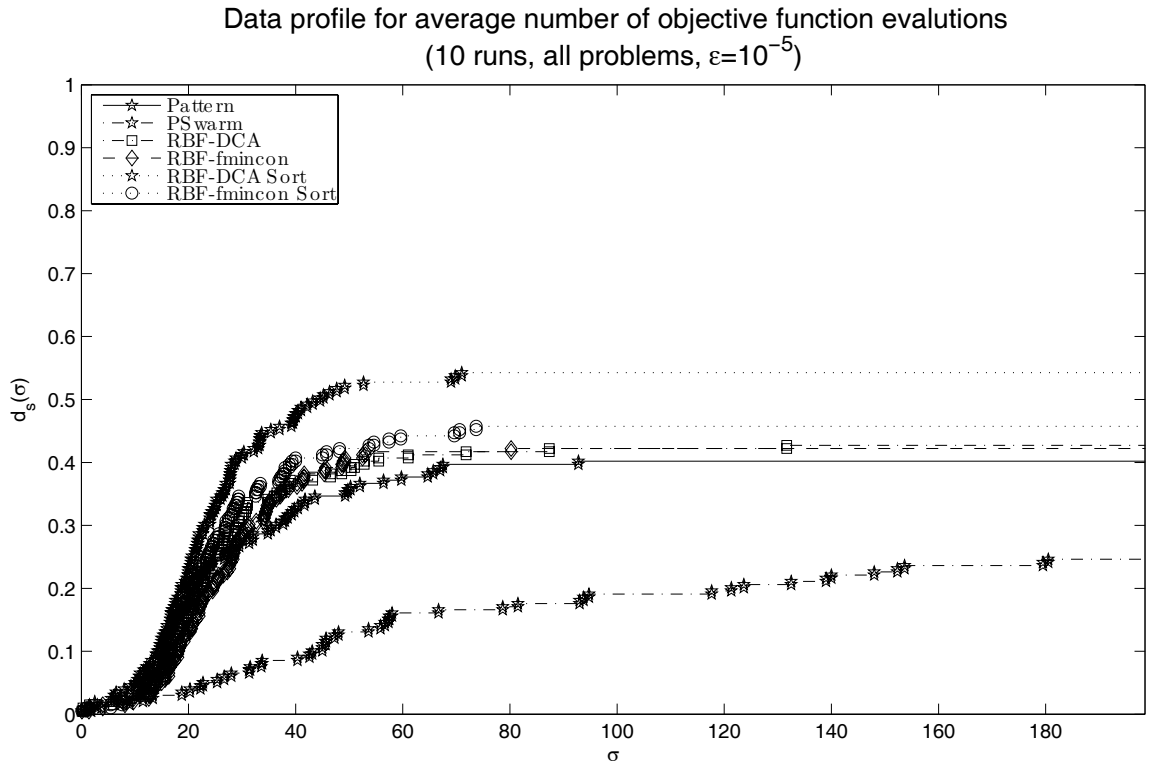

Figure 14: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for all problems (data profiles for average number of objective function evaluations).

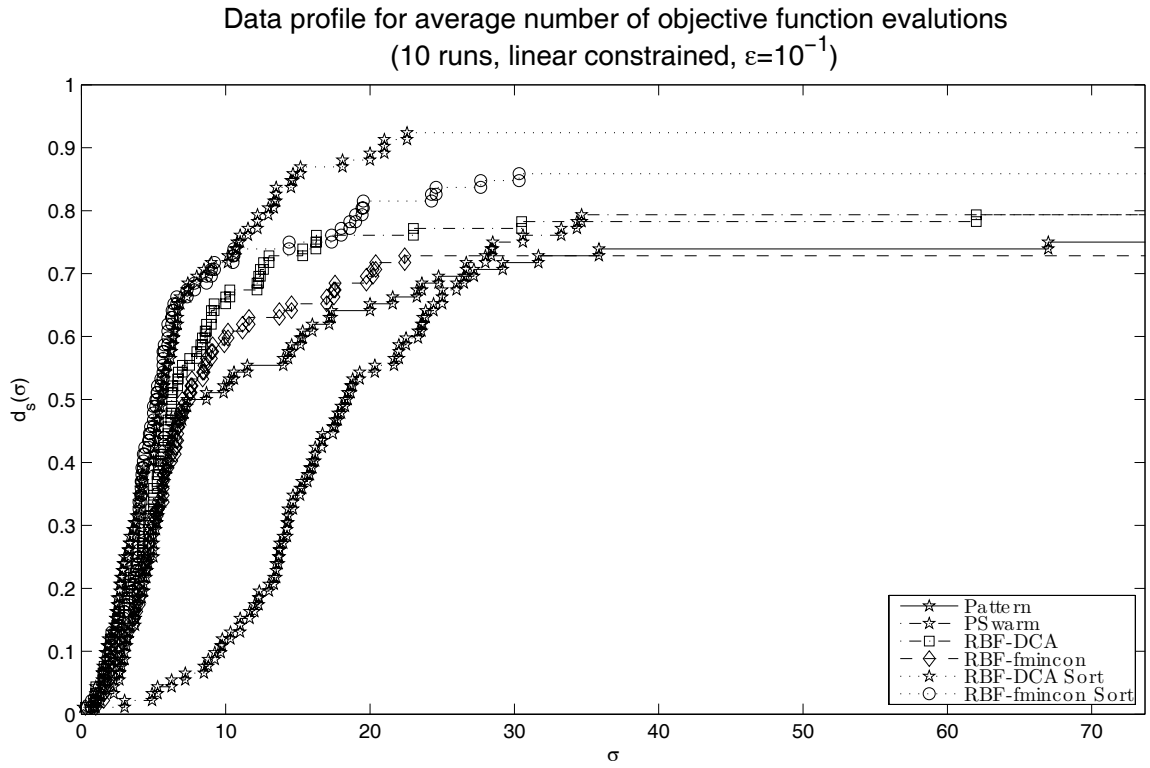

Figure 15: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for linearly constrained problems (data profiles for average number of objective function evaluations).

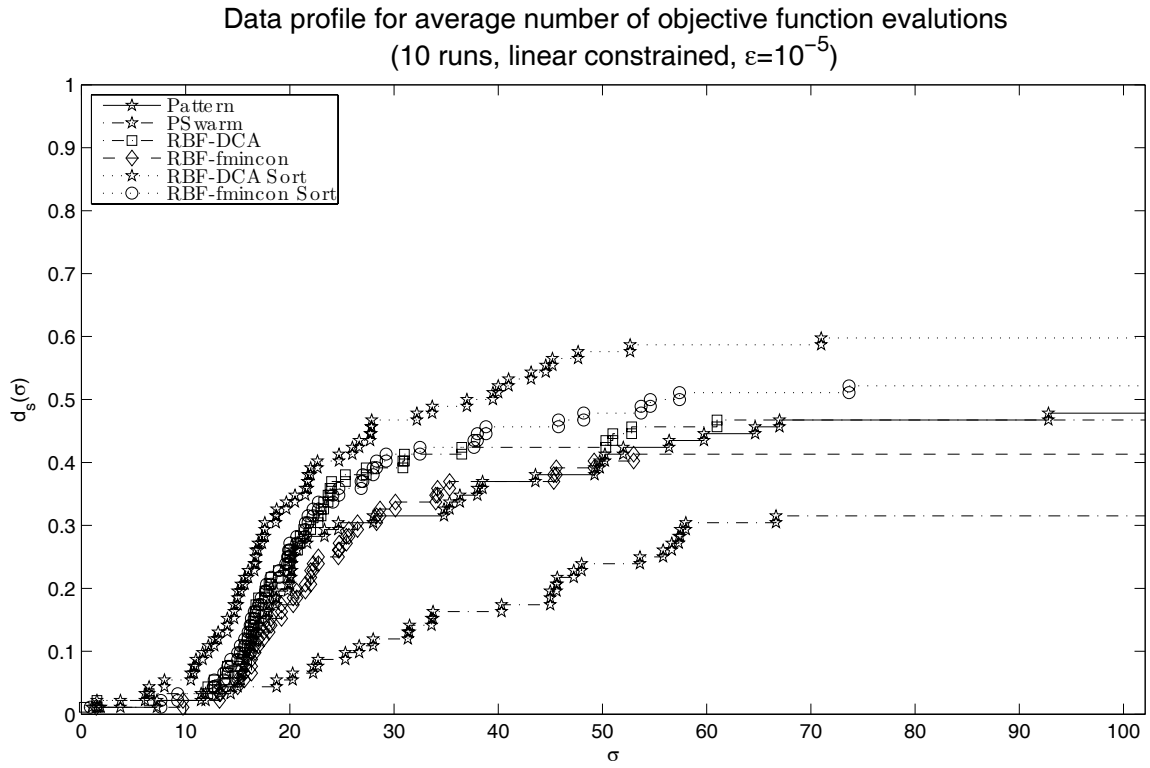

Figure 16: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for linearly constrained problems (data profiles for average number of objective function evaluations).

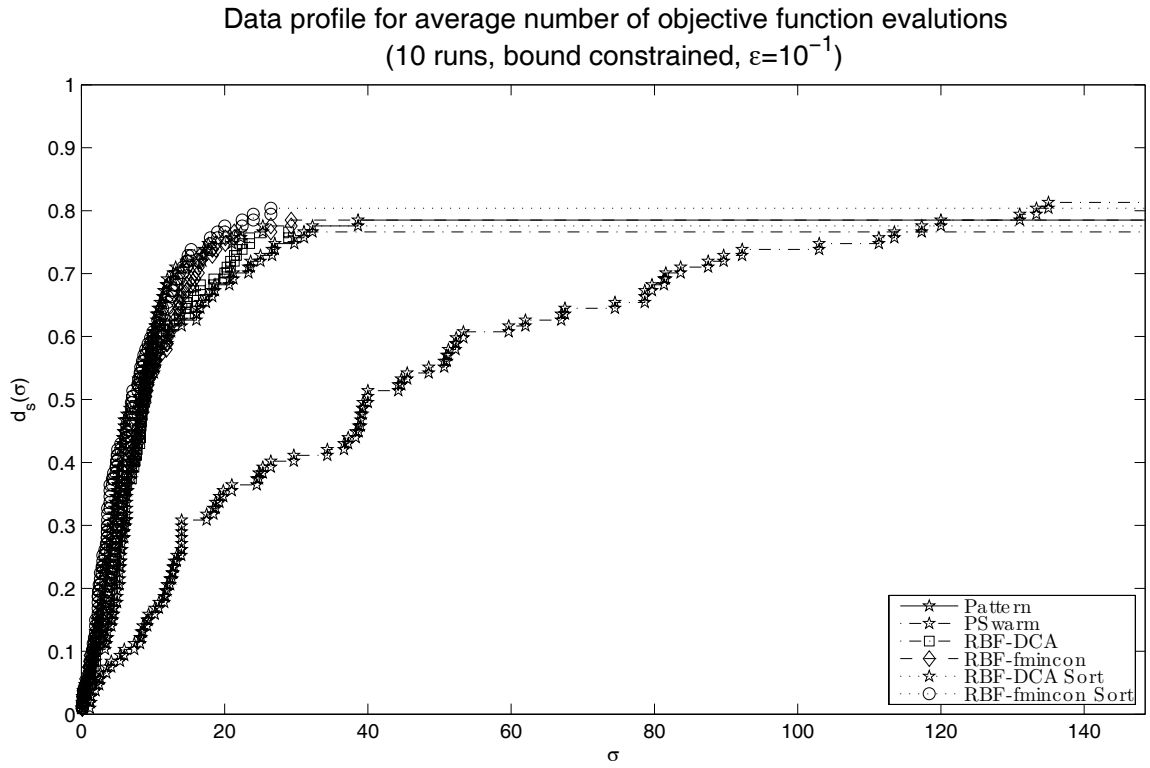

Figure 17: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for bound constrained problems (data profiles for average number of objective function evaluations).

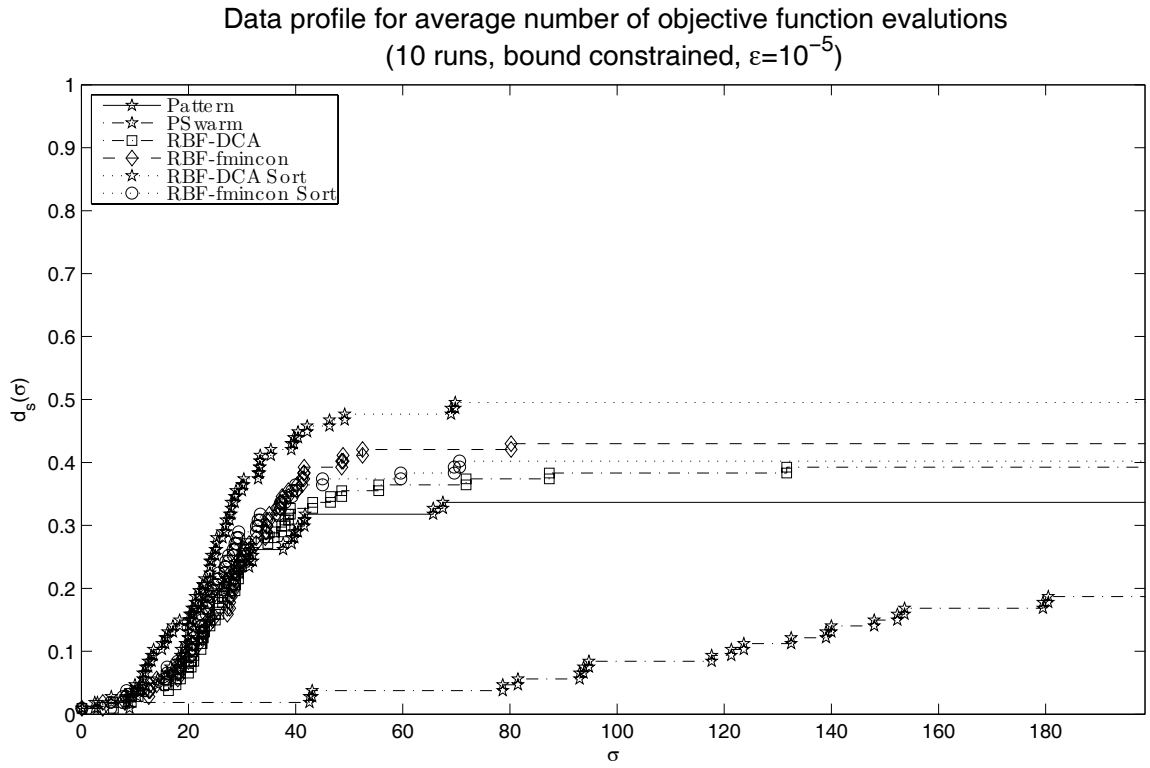

Figure 18: Comparison among RBF-DCA, RBF-DCA Sort, RBF-fmincon, RBF-fmincon Sort, Pattern, and PSwarm for bound constrained problems (data profiles for average number of objective function evaluations).

A minimizer of the RBF model is used to obtain an incumbent point where the objective function is evaluated. The RBF model minimization problem consists in a bound constrained optimization problem where the objective function (the RBF model) is cheap to evaluate. To solve the minimization problem we proposed to apply an algorithm based on difference of convex (d.c.) functions. The proposed d.c. algorithm (DCA) was compared to the MATLAB `fmincon` solver and proved to be competitive (in the sense that obtains similar objective function values using less objective function evaluations).

Extensive numerical results were reported for a test set of bound and linearly constrained problems in order to access the overall performance of the resulting direct-search derivative-free algorithms. The reported results confirmed the utility of the RBF in driving the overall direct-search algorithm for a better objective function value using less objective function evaluations.

References

[1] M. M. Ali, C. Khompatraporn, and Z. B. Zabinsky. A numerical evaluation of several stochastic algorithms on selected continuous global optimization test problems. *J. Global Optim.*, 31:635–672, 2005.

[2] Le Thi Hoai An and Pham Dinh Tao. Convex analysis approach to d.c. programming: Theory, algorithms and applications. *Acta Math. Vietnam.*, 22:289–355, 1997.

[3] Le Thi Hoai An and Pham Dinh Tao. D.C. optimization algorithms for solving the trust-region subproblem. *SIAM J. Optim.*, 8:476–505, 1998.

[4] Le Thi Hoai An and Pham Dinh Tao. 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.

[5] M. Björkman and K. Holmström. Global optimization of costly nonconvex functions using radial basis functions. *Optim. Eng.*, 1:373–397, 2000.

[6] A. R. Conn, K. Scheinberg, and L. N. Vicente. *Introduction to Derivative-Free Optimization*. MPS-SIAM Series on Optimization. SIAM, Philadelphia, 2009.

[7] T. Pham Dinh and S. Elbernoussi. Duality in d.c. (difference of convex functions) optimization. In *Subgradient Methods*, volume 84, pages 276–294. Birkhäuser, Basel, 1988.

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

[9] R. Fourer, D. M. Gay, and B. W. Kernighan. A modeling language for mathematical programming. *Management Sci.*, 36:519–554, 1990.

[10] J. D. Griffin and T. G. Kolda. Asynchronous parallel hybrid optimization combining DIRECT and GSS. *Optim. Methods Softw.*, 2009, to appear.

[11] H.-M. Gutmann. A radial basis function method for global optimization. *J. Global Optim.*, 19:201–227, 2001.

[12] A.-R. Hedar and M. Fukushima. Heuristic pattern search and its hybridization with simulated annealing for nonlinear global optimization. *Optim. Methods Softw.*, 19:291–308, 2004.

[13] L. Ingber and B. Rosen. Genetic algorithms and very fast simulated reannealing: A comparison. *Math. Comput. Modelling*, 16:87–100, 1992.

[14] Y. Ji, K.-C. Zhang, and S.-J. Qu. A deterministic global optimization algorithm. *Appl. Math. Comput.*, 185:382–387, 2006.

[15] D. Jones, C. Perttunen, and B. Stuckman. Lipschitzian optimization without the Lipschitz constant. *J. Optim. Theory Appl.*, 79:157–181, 1993.

[16] J.-E. Käck. Constrained global optimization with radial basis functions. Technical Report Research Report MdH-IMa-2004, Department of Mathematics and Physics, Mälardalen University, Västerås, Sweden, 2004.

[17] E. Kiseleva and T. Stepanchuk. On the efficiency of a global non-differentiable optimization algorithm based on the method of optimal set partitioning. *J. Global Optim.*, 25:209–235, 2003.

[18] T. G. Kolda, R. M. Lewis, and V. Torczon. Optimization by direct search: New perspectives on some classical and modern methods. *SIAM Rev.*, 45:385–482, 2003.

[19] M. Locatelli. A note on the Griewank test function. *J. Global Optim.*, 25:169–174, 2003.

[20] M. Locatelli and F. Schoen. Fast global optimization of difficult Lennard-Jones clusters. *Comput. Optim. Appl.*, 21:55–70, 2002.

[21] Z. Michalewicz. Evolutionary computation techniques for nonlinear programming problems. *International Transactions in Operational Research*, 1:223–240, 1994.

[22] Z. Michalewicz. *Genetic Algorithms+ Data Structures= Evolution Programs*. Springer, third edition, 1996.

[23] M. Mongeau, H. Karsenty, V. Rouzé, and J.-B. Hiriart-Urruty. Comparison of public-domain software for black box global optimization. *Optim. Methods Softw.*, 13:203–226, 2000.

[24] J. J. Moré and S. M. Wild. Benchmarking derivative-free optimization algorithms. *SIAM J. Optim.*, 20:172–191, `http://www.mcs.anl.gov/~more/dfo`, 2009.

[25] R. Oeuvray. *Trust-Region Methods Based on Radial Basis Functions with Application to Biomedical Imaging*. PhD thesis, Institut de Mathématiques, École Polytechnique Fédérale de Lausanne, Switzerland, 2005.

[26] R. Oeuvray and M. Bierlaire. BOOSTERS: A derivative-free algorithm based on radial basis functions. *International Journal of Modelling and Simulation*, 29:4634–4636, 2009.

[27] K. E. Parsopoulos, V. P. Plagianakos, G. D. Magoulas, and M. N. Vrahatis. Stretching technique for obtaining global minimizers through particle swarm optimization. In *Proc. Of the Particle Swarm Optimization Workshop*, pages 22–29, Indianapolis, USA, 2001.

[28] R. G. Regis and C. A. Shoemaker. Constrained global optimization of expensive black box functions using radial basis functions. *J. Global Optim.*, 31:153–171, 2005.

[29] R. G. Regis and C. A. Shoemaker. Improved strategies for radial basis function methods for global optimization. *J. Global Optim.*, 37:113–135, 2007.

[30] R. G. Regis and C. A. Shoemaker. Parallel radial basis function methods for the global optimization of expensive functions. *European J. Oper. Res.*, 182:514–535, 2007.

[31] R. G. Regis and C. A. Shoemaker. A stochastic radial basis function method for the global optimization of expensive functions. *INFORMS J. Comput.*, 19:497–509, 2007.

[32] T.P. Runarsson and X. Yao. Stochastic ranking for constrained evolutionary optimization. *IEEE Transactions on Evolutionary Computation*, 4:284–294, 2000.

[33] A. Ismael 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. Ismael F. Vaz and L. N. Vicente. Pswarm: A hybrid solver for linearly constrained global derivative-free optimization. *Optim. Methods Softw.*, 24:669–685, 2009.

[35] S. M. Wild, R. G. Regis, and C. A. Shoemaker. ORBIT: Optimization by radial basis function interpolation in trust-regions. *SIAM J. Sci. Comput.*, 30:3197–3219, 2008.

[36] S. M. Wild and C. A. Shoemaker. Global convergence of radial basis function trust-region algorithms. Technical Report Preprint ANL/MCS-P1580-0209, Mathematics and Computer Science Division, February 2009.