J Glob Optim (2013) 56:1217–1232
DOI 10.1007/s10898-012-9944-x

Constrained derivative-free optimization on thin domains

J. M. Martínez · F. N. C. Sobral

Received: 30 August 2011 / Accepted: 11 June 2012 / Published online: 4 July 2012

Abstract Many derivative-free methods for constrained problems are not efficient for minimizing functions on "thin" domains. Other algorithms, like those based on Augmented Lagrangians, deal with thin constraints using penalty-like strategies. When the constraints are computationally inexpensive but highly nonlinear, these methods spend many potentially expensive objective function evaluations motivated by the difficulties in improving feasibility. An algorithm that handles this case efficiently is proposed in this paper. The main iteration is split into two steps: restoration and minimization. In the restoration step, the aim is to decrease infeasibility without evaluating the objective function. In the minimization step, the objective function f is minimized on a relaxed feasible set. A global minimization result will be proved and computational experiments showing the advantages of this approach will be presented.

Keywords Derivative-free optimization · Disconnected domains · Global convergence · Numerical experiments · Thin domains

1 Introduction

In many practical problems one needs to minimize functions whose derivatives are not available for several reasons. The recent book by Conn et al. [13] surveys the most relevant existing approaches for the unconstrained case and predicts that much research should be expected with regards to constrained problems in forthcoming years. Unconstrained techniques based

This work was supported by PRONEX-CNPq/FAPERJ Grant E-26/171.164/2003-APQ1, FAPESP Grants 03/09169-6, 06/53768-0 and 08/00468-4, and CNPq.

J. M. Martínez · F. N. C. Sobral (✉)
Department of Applied Mathematics, Institute of Mathematics, Statistics and Scientific Computing, University of Campinas, Campinas, SP, Brazil
e-mail: fsobral@ime.unicamp.br

J. M. Martínez
e-mail: martinez@ime.unicamp.br

on local explorations, line searches or quadratic models [11,14,36] can be suitably adapted to box-constrained and linearly constrained derivative-free optimization [20–22,34]. Mesh adaptive direct search methods [3,5] are very effective for practical problems in which a "robust" feasible set may be efficiently explored by means of local samples that do not involve function evaluations at infeasible points. Problems with more general constraints were addressed by means of Augmented Lagrangian approaches in [15,32] and [18,23,25]. In [23], the Augmented Lagrangian method is based on the Lancelot approach [10], whereas in [15] and [25] the authors use the Algencan framework [1]. (See www.ime.usp.br/~egbirgin/tango.) In [15] two types of constraints are considered: the "difficult" ones are included in the augmented formulation of the Lagrangian, whereas "easy" constraints remain in the subproblems that are solved at each outer iteration. In this way, subproblems may be effectively solved by specific algorithms or by algorithms based on the MADS [3,24] approach. Easy constraints may be sometimes identified with the "non-relaxable" constraints considered in [4].

The Augmented Lagrangian approach imposes the evaluation of the objective function and the (difficult) constraints at the same (perhaps infeasible) points. This is not convenient when the objective function is very expensive to compute and the current point is infeasible. In this case we would like to restore feasibility without spending objective function evaluations, a resource that is not available in Augmented Lagrangian methods.

On the other hand, methods that maintain feasibility of all the iterates, such as MADS-like and barrier methods, cannot work well in the presence of nonlinear equality constraints, or nonlinear very thin domains.

This state of facts motivated us to revisit a traditional approach for derivative-free nonlinear programming. In 1969, Paviani and Himmelblau [31] adapted the simplex Nelder-Mead method [30] for constrained optimization. Their iterative method employs a decreasing tolerance for infeasibility that depends on the simplex size. Whenever a trial point violates that tolerance, it is replaced by a restored point whose feasibility violation can be admitted. Clearly, this procedure produces severe simplex deformations which impair its chances of practical convergence. However, the idea of using decreasing infeasibility tolerances is quite valuable and has been employed in several modern methods for constrained (not necessarily derivative-free) optimization [7,9,17,26,27].

We will say that the feasible domain of an optimization problem is "thin" if at least one of the following two properties hold:

1. For a considerable number of feasible points, say x, given any direction d and any scalar $\alpha > 0$, the probability that $x + \alpha d$ is feasible is very small (perhaps zero).
2. The domain is disconnected, which means that it can be separated at least into two nonempty subsets, each one belonging to a different disjoint open set.

Observe that the second property is, in some sense, a limit case of the first one, since a disconnected domain may be connected by means of a finite number of segments, which are obviously thin.

Except in pathological cases, a single equality constraint is enough to define a thin domain. In general, we will focus on problems in which:

1. Derivatives of the objective function are not available. Gradients of the constraints may be available or not.
2. The evaluation of the objective function is computationally expensive, but the constraints are easy to evaluate.
3. Although the constraints are inexpensive, they define a domain for which finding a feasible point may require several (cheap) numerical computations.

At each iteration of the algorithm introduced in this paper, we define a relaxed "fat" domain on which we (approximately) minimize the objective function using an algorithm that, presumably, handles efficiently these relaxed cases. Here we use "fat" as a synonym of "not-thin". We use a restoration procedure that does not involve the objective function for initializing the subproblem resolution. The infeasibility tolerance for the relaxed domain tends to zero as long as iterations advance. The combination of objective-function-free restorations and derivative-free relaxed minimizations produces a sequence that converges to the solution of the original problem, under reasonable algorithmic conditions.

The main algorithm will be described in Sect. 2, where we give a global convergence result. In Sect. 3 we describe a computer implementation. In Sect. 4 we show numerical experiments. Section 5 is devoted to conclusions of the present work.

Notation We denote $\mathbb{N} = \{0, 1, 2, \ldots\}$.

We define $\mathbb{R}_+ = \{t \in \mathbb{R} \mid t \geq 0\}$ and $\mathbb{R}_{++} = \{t \in \mathbb{R} \mid t > 0\}$.

$\mathscr{B}(x, \Delta)$ denotes the closed ball with center x and radius Δ, with respect to a given norm $\|\cdot\|$.

2 Algorithms

The problem under consideration in this paper is:

$$\text{Minimize } f(x) \text{ subject to } g(x) \leq 0 \text{ and } x \in \Omega, \tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$, $g : \mathbb{R}^n \to \mathbb{R}^p$ and the closed set $\Omega \subseteq \mathbb{R}^n$ represents the non-relaxable constraints in the sense of [4]. Equality constraints of the form $h(x) = 0$ are supposed to be replaced in (1) with the two obvious inequalities $h(x) \leq 0$ and $-h(x) \leq 0$. This replacement causes an increase in the total number of constraints, but does not affect the convergence of the algorithm. The numerical aspects remain unchanged because we suppose that the evaluation of the constraints is inexpensive. In Sect. 4 we solve a set of equality-constrained test problems and compare with algorithms that deal explicitly with the case $h(x) = 0$.

A high-level description of the main algorithm is given below. The lack of details in Algorithm 1.1 allows one to use efficient methods which exploit the special structure of the problems. An implementation of each step is fully given in Sect. 3.

In the rest of the paper, given $w^k \in \mathbb{R}^p_+$, a point x that satisfies

$$g(x) \leq w^k \quad \text{and} \quad x \in \Omega \tag{2}$$

will be called w^k-feasible.

Algorithm 1.1 Let $x^0 \in \Omega$ be an initial approximation. Let $w^1 \in \mathbb{R}^p_+$. Set $k \leftarrow 1$.

Step 1 (Restoration)
Compute a w^k-feasible point y^k (Any strategy can be used, in order to find y^k).

Step 2 (Minimization)
Using Algorithm 1.2 below, compute a w^k-feasible "approximate solution" x^k of the following subproblem:

$$\text{Minimize } f(x) \text{ subject to } g(x) \leq w^k \text{ and } x \in \Omega. \tag{3}$$

Step 3 (Updating)
Choose $w^{k+1} \in \mathbb{R}^p_+$, set $k \leftarrow k + 1$ and go to Step 1.

Note that the restoration step plays no role in the formal definition of the algorithm, since we suppose that the minimization step finds a w^k-feasible "approximate solution". The restored point y^k will be used, in practice, as starting point for solving the subproblem (3). Feasible starting points are necessary when we use derivative-free algorithms based on feasible iterates. (like coordinate search, GSS [20] and standard MADS [3]) The definition of w^{k+1} is not given in Algorithm 1.1. However, since w^k is related to the level of allowed infeasibility, we actually need $\{w^k\}$ converging to 0 in order to find feasible points.

Algorithm 1.2 describes the way in which x^k is computed at Step 2 of Algorithm 1.1 and, in fact, defines what we mean by "approximate solution" of the subproblem (3). Again, we present a high-level description of the algorithm here. Practical details are given in Sect. 3.

At Step 2 of Algorithm 1.2 we use an operator φ_k. This operator is applied to a direction v and evokes the projection of the point $x + v$ on the relaxed domain. Thus, if $x + v$ is w^k-infeasible, the point $x + \varphi_k(v)$ is w^k-feasible. The operator φ_k is not known in advance, and is evaluated at Step 2.2 employing a restoration procedure similar to the restoration step of Algorithm 1.1. In Sect. 3 we show one possible way of performing this restoration.

Algorithm 1.2 (Minimization step at Algorithm 1.1)

Assume that $\beta \geq 1$, $\Delta > 0$, $\eta_k > 0$, m_k is a given positive integer and y^k comes from the restoration Step 1 of Algorithm 1.1. Define $z^0 = y^k$ and set $\ell \leftarrow 0$.

Step 1 (Improvement step)

Find $z \in \Omega$ such that $g(z) \leq w^k$ and $f(z) \leq f(z^\ell)$. Note that $z = z^\ell$ is a possible choice, but, in practice, we suggest using an heuristic or a well-established derivative-free algorithm to solve the relaxed "fat" subproblem (3).
Set $z^{\ell+1} = z$ and $\ell \leftarrow \ell + 1$.

Step 2 (Poll step)
Set $V_k \leftarrow \emptyset$, $j \leftarrow 1$ and choose $\eta_{k,\ell} \in [\eta_k, \beta\eta_k]$.

Step 2.1 Compute a "pseudo-random" direction $v \in \mathscr{B}(0, \Delta)$.
(The more usual way is to use the infinity norm to define $\mathscr{B}$, employing a pseudo-random uniform generator, like [35], to generate points in the interval $[-\Delta, \Delta]^n$). We describe a different way in Sect. 3.

Step 2.2 If $z^\ell + v$ is w^k-feasible, define $\varphi_k(v) = v$ and go to Step 2.3. Else, try to find, by means of restoration, a new direction $\varphi_k(v)$ such that $z^\ell + \varphi_k(v)$ is w^k-feasible. If this procedure is successful, go to Step 2.3. Else, go to Step 2.4.

Step 2.3 If

$$f(z^\ell + \varphi_k(v)) < f(z^\ell) - \eta_{k,\ell}\,, \tag{4}$$

define $z^{\ell+1} = z^\ell + \varphi_k(v)$, set $\ell \leftarrow \ell + 1$ and go to Step 1. Else, go to Step 2.4.

Step 2.4 Set $V_k \leftarrow V_k \cup \{v\}$.
If $j = m_k$, set $x^k = z^\ell$ and return to Step 3 of Algorithm 1.1. Else, set $j \leftarrow j + 1$ and go to Step 2.1.

At Step 1 of Algorithm 1.2 one tries to improve the objective function value in the region defined by $g(x) \leq w^k, x \in \Omega$. The idea will be to use a well-established derivative-free

optimization method for this purpose, trying to assure that x^k is w^k-feasible. Note that the relaxed subproblem (3) is not thin-constrained and can be solved by several algorithms described in literature [3,4,20]. This step is not essential for proving global convergence, which is based on the procedure adopted at Step 2. At Step 2 one chooses an arbitrary "pseudo-random" point on the ball with radius Δ centered on the current w^k-feasible point and we "project" this point on the w^k-feasible region. If, after m_k consecutive trials, the objective function fails to decrease enough or restoration cannot be completed, the algorithm returns with the last iterate computed by Algorithm 1.2. The following assumptions guarantee that Algorithm 1.1, with Algorithm 1.2 at the minimization step, finds global minimizers within a ball of radius Δ. If Ω is bounded and Δ is large enough, this means that global minimizers of the original problems are found up to any arbitrary precision.

Assumption P1 The functions f and g are continuous.

Assumption P2 For all $w \geq 0$ the function f is bounded below on the set defined by $x \in \Omega$ and $g(x) \leq w$.

Under Assumptions P1 and P2, it is easy to see that Algorithm 1.2 performs a finite number of iterations. Otherwise, we would have an infinite sequence such that $f(z^{\ell+1}) < f(z^\ell) - \eta_{k,\ell} \leq f(z^\ell) - \eta_k$ for all ℓ and, thus, $\lim_{\ell\to\infty} f(z^\ell) = -\infty$. Moreover, Assumption P2 is not restrictive, since it may be guaranteed by the addition of bound constraints on the original problem.

Assumption A1 The sequences $\{\|w^k\|\}$ and $\{\eta_k\}$ tend to zero.

Assumption A2 If $\{x^k\}_{k\in K}$ is a convergent subsequence generated by Algorithm 1.1, the set V, defined by

$$V = \bigcup_{k\in K} V_k,$$

is dense in $\mathscr{B}(0, \Delta)$.

Assumption A2 resembles the density requirements of the MADS framework but makes no use of dense directions in the unitary sphere lying in integer lattices. Our approach is closely connected to the simple framework described in [37], where the use of sufficient decrease and dense directions in the unitary sphere avoided the necessity of integer lattices.

Assumption A3 If $\{x^k\}_{k\in K}$ is a convergent subsequence generated by Algorithm 1.1, $v^k \in V_k$ for all $k \in K$, and $\lim_{k\in K} x^k + v^k = z$, where $g(z) \leq 0$ and $z \in \Omega$, then, for all $k \in K$ large enough, the restoration procedure that computes $\varphi_k(v^k)$, at Step 2.2 of Algorithm 1.2, is successful.

According to Step 2 of Algorithm 1.2, the directions of V_k are of two types: those associated with a successful calculation of φ_k and those associated with a failure. When the restoration procedure for a w^k-infeasible point $z^\ell + v$ fails, condition (4) is not verified and, therefore, there is no information about $f(z^\ell + \varphi_k(v))$. If this lack of information occurred at infinitely many iterations of Algorithm 1.1, the set V_k would contain several directions that could not be used as poll directions, due to the w^k-infeasibility. Thus, we would not be able to guarantee optimality of limit points. Assumption A3 is necessary to ensure that the algorithm has really polled all the directions of the sets V_k, for all sufficiently large k. To have this property, the process that constructs φ_k cannot fail. Numerical experiments show that Assumption A3 is usually satisfied.

Assumption A4 If $\{x^k\}$, $\{v^k\}$, K, and z are as in Assumption A3, then

$$\lim_{k\in K} x^k + \varphi_k(v^k) = z.$$

It is known that the orthogonal projection is a continuous function, when specific types of sets are considered. Since we construct the operator φ_k to act as a projection operator, what Assumption A4 says is that φ_k has to resemble the continuity property of an orthogonal projector. In fact, if the set $\{x \in \Omega \mid g(x) \leq w^k\}$ is convex and $z^\ell + \varphi_k(v)$ is the projection of $z^\ell + v$ on this set, then the distance between $z^\ell + \varphi_k(v)$ and z will not be bigger than the distance between $z^\ell + v$ and z, for all feasible z. This will be enough to guarantee the fulfillment of Assumption A4. This observation suggests that a reasonable way to compute $\varphi_k(v)$ at Step 2.2 of Algorithm 1.2, consists of minimizing the distance between $z^\ell + v$ and the feasible set. When k is large and $z^\ell + v$ is close to the feasible set, it is reasonable to assume that a suitable algorithm not employing functional evaluations will be able to find a projection-like restored point that approximates $z^\ell + v$ to the feasible region, as required by Assumption A3.

Theorem 2.1 *Assume that $\{x^k\}$ is generated by Algorithm 1.1, Ω is closed, Assumptions P1–P2 and A1–A4 hold, and x^* is a limit point of the sequence $\{x^k\}$. Then, x^* is a global solution of the problem*

$$\textit{Minimize } f(x) \textit{ subject to } g(x) \leq 0,\ x \in \Omega \textit{ and } \|x - x^*\| \leq \Delta.$$

Proof By the hypothesis, $g(x^k) \leq w^k$ and $x^k \in \Omega$ for all k. Then, by Assumptions P1, A1 and the closedness of Ω, we have that $g(x^*) \leq 0$ and $x^* \in \Omega$.

Assume now that $\lim_{k\in K} x^k = x^*$. Let $z \in \Omega$ be such that

$$\|z - x^*\| \leq \Delta$$

and $g(z) \leq 0$. Since $z - x^* \in \mathscr{B}(0, \Delta)$, by Assumption A2 and the finiteness of V_k, there exists an infinite set of indices $K_1 \subseteq K$ and a sequence $\{v^k\}_{k\in K_1}$ such that $\lim_{k\in K_1} v^k = z - x^*$ and $v^k \in V_k$ for all $k \in K_1$. Thus, $\lim_{k\in K_1} x^k + v^k = z$. By Assumption A3, $\varphi_k(v^k)$ is well defined if k is large enough. By Assumption A4, this implies that $\lim_{k\in K_1} x^k + \varphi_k(v^k) = z$.

By Assumption A2, Algorithm 1.2 returns with the point x^k and, for all $v \in V_k$, x^k is such that condition (4) does not hold. Then, for $v^k \in V_k$, $k \in K_1$ large enough,

$$f(x^k + \varphi_k(v^k)) \geq f(x^k) - \eta_{k,\ell} \geq f(x^k) - \beta\eta_k.$$

Taking limits, by the continuity of f we have that $f(z) \geq f(x^*)$. □

Remarks Since the famous anonymous paper [2], we know that one should be cautious about the interpretation of convergence results that use "density arguments". These results could be insufficient to explain efficiency and robustness of the methods to which they are applied. Moreover, the rigorous implementations of algorithms that can be guaranteed to satisfy density requirements could be unacceptably expensive, since full generation of dense sets demands a computational work that grows exponentially with the number of variables. However, in the global optimization framework, if one does not use specific information about the problem, only dense-set arguments can guarantee theoretical convergence to global optimizers. For similar reasons, dense-set arguments became popular for analyzing convergence of constrained derivative-free methods in the last few years. In general, density arguments have been used for proving convergence to different types of stationary points in this context [4,5,13].

In this section we proved that the Algorithm 1.1 obtains local minimizers within a fixed neighborhood of arbitrary radius $\Delta > 0$. If Δ is large enough and the feasible set is bounded, this means that the algorithm finds global optimizers of the original problem. In spite of the drawbacks pointed out above, the proof of Theorem 2.1, as many other density-based proofs, sheds some light on the characteristics that a practical algorithm should exhibit in order to be reasonably efficient.

Assume that x^* is a limit point of a sequence generated by Algorithm 1.1 and that z is a feasible point such that $\|z - x^*\| \leq \Delta$. The proof of Theorem 2.1 shows that, in order to obtain the desired property $f(z) \geq f(x^*)$ it is necessary to guarantee that, for k large enough, w^k-feasible points arbitrarily close to z are tested by the algorithm used to solve the subproblems. This property is guaranteed by the assumptions of the theorem, but, in practical terms, we want to satisfy it spending a moderate amount of computer work. This practical requirement prescribes that, at least for k large enough ($\|w^k\|$ small enough), $f(x^k)$ should be smaller than $f(x)$ for all x in a "not very small" set of w^k-feasible points in the ball $\mathscr{B}(x^k, \Delta)$. As a consequence of being "not very small", the probability that points in this set approximate any arbitrary feasible point z such that $\|z - x^*\| \leq \Delta$, may be significant as k grows (although, obviously, this probability decreases dramatically with n). Now, when $\|w^k\|$ is small, the probability of generating feasible points using, say, random generation or grids, may be very small. However, one of the assumptions of our problem is that constraints are cheap and only the objective function is expensive. Therefore, given possibly infeasible (random or grid) generated points, cheap restoration procedures may be used to generate close feasible ones.

Algorithm 1.2 may be implemented without the improvement step (equivalently, taking $z = z^\ell$ at Step 1). However, we wish to take advantage of the fact that several existing algorithms are efficient for solving (3) if $\|w^k\|$ is not excessively small. In our implementation, we will use DFO [11,12] for finding z at the first step of Algorithm 1.2.

3 Implementation

This section discusses implementation details regarding the algorithms of Sect. 2. Algorithm 1.1 has been implemented using Algorithm 1.2 at the minimization step (Step 2).

3.1 Restoration step

In the restoration step (Step 1 of Algorithm 1.1), in order to find the point y^k we consider the following auxiliary problem

$$\text{Minimize } \|y - x^k\|_2^2 \text{ subject to } g(y) \leq \tau w^k \text{ and } y \in \Omega, \tag{5}$$

where $\tau \in [0, 1]$ is a given algorithmic parameter that controls the restoration effort. The point y^k is an approximate solution of (5). For solving this subproblem we use Algencan [1] with its default parameters, if constraint derivatives are available. The standard coordinate search algorithm is used when derivatives of g are not available. In both cases, we assume that the non-relaxable constraints define a box. If one fails to find an w^k-feasible point in the process of solving (5), we continue the execution of Algorithm 1.1 at Step 2 taking the (perhaps infeasible) output of (5) as initial approximation. Since the restored point y^k plays no role in the formal definition of the algorithm, as mentioned in Sect. 2, this does not affect the convergence proof.

3.2 Improvement step

In the improvement step (Step 1 of Algorithm 1.2) we use DFO [11], a derivative-free trust-region algorithm for unconstrained problems which is also able to handle general smooth constraints. When dealing with constraints, DFO acts as an heuristic method that computes only feasible points and, at each iteration, minimizes a quadratic model of the objective function subject to the problem's constraints and a suitable trust region [12]. For solving the smooth constrained trust-region subproblems required in DFO, we use Algencan. Note that DFO can be applied to the original problem (1). In fact, we will apply DFO directly to (1) for numerical comparisons in Sect. 4.

3.3 Pseudo-random directions

At Step 2 of Algorithm 1.2, we compute the direction v as a pseudo-random vector with uniform distribution in $\mathscr{B}(0, \Delta)$. For this purpose we use the method of Muller [29] to generate a pseudo-random vector $v' \in \mathbb{R}^n$ uniformly distributed in the unitary hypersphere. Then, we choose a pseudo-random number $\rho \in [0, 1]$ by means of Schrage's algorithm [35]. The vector $v = (\rho^{1/n}\Delta)v'$ turns out to be a pseudo-random vector uniformly distributed in the Euclidean ball $\mathscr{B}(0, \Delta)$ [6].

3.4 Computation of φ_k

Similarly to (5), we compute $\tilde{v} = \varphi_k(v)$ as an approximate solution of

$$\text{Minimize } \|\tilde{v} - v\|_2^2 \text{ subject to } g(z^\ell + \tilde{v}) \leq w^k \text{ and } z^\ell + \tilde{v} \in \Omega. \tag{6}$$

In this way, $z^\ell + \varphi_k(v)$ has the flavor of a projection of $z^\ell + v$ on the feasible set, as required by the theoretical Assumption A4.

For solving (6) we proceed as in the subproblem (5). The case in which restoration is not successful is covered by Step 2.2 of Algorithm 1.2.

3.5 Further implementation features

Tolerances, stopping criteria and other implementation details are given below. We employ the usual norms: $\|\cdot\|_\infty$ (the infinity norm) and $\|\cdot\|_2$ (the Euclidean norm).

1. At Algencan executions for solving (5) and (6) we use x^k and v as initial points, respectively. We employ the default parameters of Algencan, using 10^{-8} as stopping criteria both for feasibility and optimality.
2. The rule for updating the tolerances w^k is as follows. Initially, we define

$$w_i^1 = \max\{10, \|g(x^0)\|_\infty\}, \quad \forall i = 1, \ldots, p. \tag{7}$$

At the end of each iteration of Algorithm 1.1 we force the point x^k, that comes from the minimization step, to be w^{k+1}-infeasible, taking:

$$w_i^{k+1} = \max\{10^{-8}, \theta \min\{w_i^k, \|g(x^k)\|_\infty\}\}, \quad \forall i = 1, \ldots, p, \tag{8}$$

where $\theta \in (0, 1)$. In the experiments reported in this paper we used $\theta = 0.1$.

3. The objective function used in the improvement step by DFO has a fixed penalization of the infeasibility:

$$\bar{f}(x) = f(x) + \rho \|g(x)_+\|_2^2,$$

where $g(x)_+ = \max\{0, g(x)\}$ (in vector form) and

$$\rho = \max\{1, |f(x^0)|\} / \max\{1, \|g(x^0)\|_\infty\} \tag{9}$$

is the penalty parameter.
Note that the parameter ρ is fixed and that the problem of minimizing $\bar{f}(x)$ subject to $g(x) \leq 0$ and $x \in \Omega$ is equivalent to the original problem (1), so, there is no loss of generality when we consider $\bar{f}(x)$ instead of the original $f(x)$ in the algorithm. Since we assume that the constraints are cheap and ρ is fixed, there is no loss of stability and no significative increase of computer time can be attributed to this replacement.
4. We employ the default parameters of DFO in the improvement step. The modified stopping criteria were: trust region radius smaller than $\min\{10^{-1}, \max\{10^{-3}, \|w^k\|_\infty^2\}\}$, maximum of $1,000n$ function evaluations and maximum of $1,000$ iterations. In the constrained trust-region subproblems solved by Algencan in the context of DFO, we use 10^{-8} as stopping criteria for both optimality and feasibility.
5. The objective function used in the poll step is the extreme barrier function [3] with fixed penalization of the infeasibility:

$$\tilde{f}(x) = \begin{cases} f(x) + \rho \|g(x)_+\|_2^2, & \text{if } x \text{ is } w^k\text{-feasible} \\ 10^{99}, & \text{otherwise,} \end{cases} \tag{10}$$

where the parameter ρ is defined by Eq. (9).
6. The sufficient decrease parameter η_k is defined as $\eta_k = 10^{-8}/k$, for $k = 1, 2, \ldots$. The parameter β is set to 10^{16}. For each $\ell = 1, 2, \ldots$, in Algorithm 1.2 we use

$$\eta_{k,\ell} = \max\{\eta_k, \min\{\|g(z^\ell)\|_\infty, \eta'_k, \beta\eta_k\}\}, \tag{11}$$

where η'_k is defined as follows:

$$\eta'_k = \begin{cases} 0.01 \max\{1, |f(x^0)|\}, & \text{if } k = 1 \\ \min\{0.5\eta'_{k-1}, \|w^k\|_\infty\}, & \text{if } k > 1. \end{cases}$$

Equation (11) guarantees that $\eta_{k,\ell} \in [\eta_k, \beta\eta_k]$, as required by Algorithm 1.2, and the definition of η_k guarantees that $\eta_k \to 0$. So, we ask for a substantial decrease in the poll step, unless we already have a feasible point. The term η'_k adds information about the parameter w^k in $\eta_{k,\ell}$. The idea is that we are leaving the main work to the improvement step (DFO) at the first iterations and when x^k becomes almost feasible, the poll step really acts.
7. We choose $m_k = 2n$ for all k and also set $\Delta \equiv 1$. The parameter Δ controls the radius of the search in the poll step and has to be chosen according to the local optimality desired for x^*. Our choice showed a good trade-off between optimality and computational effort.
8. We use $\tau = 0.1$ in (5).
9. The algorithm declares success when, at the end of the minimization step, it finds x^k satisfying $g(x^k) \leq 10^{-8}$. It declares failure when $\|w^k\|$ becomes smaller than 10^{-8} and the current point x^k is not w^k-feasible or when the algorithm reaches the maximum of 10^3 iterations or the maximum of 10^6 functional evaluations.

Table 1 Description of the test problems

Prob.	Var.	Ineq.	Eq.	Prob.	Var.	Ineq.	Eq.	Prob.	Var.	Ineq.	Eq.
6	2	0	1	49	5	0	2	74	4	2	3
7	2	0	1	50	5	0	3	75	4	2	3
8	2	0	2	51	5	0	3	77	5	0	2
9	2	0	1	52	5	0	3	78	5	0	3
14	2	1	1	53	5	0	3	79	5	0	3
26	3	0	1	54	6	0	1	80	5	0	3
27	3	0	1	55	6	0	6	81	5	0	3
28	3	0	1	56	7	0	4	87	6	0	4
32	3	1	1	60	3	0	1	99	7	0	2
39	4	0	2	61	3	0	2	107	9	0	6
40	4	0	3	62	3	0	1	109	9	4	6
41	4	0	1	63	3	0	2	111	10	0	3
42	4	0	2	68	4	0	2	112	10	0	3
46	5	0	2	69	4	0	2	114	10	8	3
47	5	0	3	71	4	1	1	119	16	0	8
48	5	0	2	73	4	2	1				

4 Numerical experiments

We compared our algorithm against the C++ implementation of an Augmented Lagrangian derivative-free algorithm [25], called HOPSPACK [33], and the Fortran implementation of DFO. From now on, Algorithm 1.1, with the implementation described in the previous section, will be called *Skinny*.

We employed 47 problems from the Hock and Schittkowski collection [19] in our tests. This test set is a well-known collection of 116 small problems commonly used to test derivative-free algorithms. We selected only the problems that exhibit thin domains (i.e. having at least one equality constraint), since *Skinny* was designed to handle this case.

The derivatives of the constraints were used by DFO and by Algencan in the restoration problems (5) and (6). *Skinny* was implemented in Fortran 90 and DFO in Fortran 77. Both were compiled with `gfortran-4.2` and ran on a computer with 8GB of RAM, two Intel Core i7 2.67GHz processors and 64bit Linux (Ubuntu) operational system. The problems considered in this study are reported in Table 1, using their numbers in [19]. In this table, Var denotes the number of variables, Ineq represents the number of inequality constraints, and Eq represents the number of equality constraints.

For algorithmic comparisons we used data and performance profiles as described in [16,28]. We will consider that an algorithm has successfully converged if the obtained solution $\bar{x}$ is feasible and:

$$\frac{|f(\bar{x}) - f_{\mathrm{L}}|}{\max\{1, |f(\bar{x})|, |f_{\mathrm{L}}|\}} \leq 10^{-1}, \tag{12}$$

where f_{L} is the best value of f among the feasible solutions found by the compared algorithms. The number of function evaluations was used as performance measure.

For the numerical comparisons, we required a feasibility tolerance of 10^{-8} in the three algorithms. In Algencan we employed the default algorithmic parameters. Table 2 shows the

Table 2 Comparison between *Skinny*, HOPSPACK and DFO in 47 problems having thin domains

Prob.	*Skinny*		HOPSPACK		DFO	
	f	#FE	f	#FE	f	#FE
6	8.8304E−08	97	4.8400E+00*	151	2.2009E−23	14
7	−1.7321E+00	150	6.9315E−01	325	−1.7321E+00	10
8	−1.0000E+00	56	−1.0000E+00*	187	−1.0000E+00	3
9	−5.0000E−01	109	−5.0000E−01	26	−5.0000E−01	12
14	1.3935E+00	142	1.3941E+00	202	1.3935E+00	9
26	2.1160E+01	33	2.1160E+01	585	1.2942E−06	40
27	4.0000E+00	269	4.0006E+00*	1,358	4.0000E+00	33
28	3.6892E−27	43	7.7034E−08	264	2.3039E−18	25
32	1.0000E+00	209	1.0000E+00	51	1.0000E+00	11
39	−1.0000E+00	302	−1.0000E+00	830	−1.0000E+00	23
40	−2.5000E−01	215	−2.5056E−01*	897	−2.5000E−01	14
41	1.9259E+00	348	1.9259E+00	292	1.9259E+00	34
42	1.3858E+01	254	1.4000E+01	779	1.3858E+01	14
46	2.1090E−02	501	3.3376E+00	777	2.9829E−07	77
47	1.4548E−08	302	1.2495E+01	901	1.8714E−06	50
48	1.4969E−16	76	1.1174E−06	497	1.2711E−18	29
49	3.7388E−05	261	1.4294E−04	1,002	2.7801E−05	72
50	7.7030E−06	246	5.2937E−07	290	2.1369E−07	46
51	8.4671E−17	88	1.2537E−06	142	1.1025E−18	16
52	5.3267E+00	337	5.3267E+00*	311	5.3266E+00	20
53	4.0930E+00	295	4.0930E+00	216	4.0930E+00	18
54	−1.5581E−01	335	–	–	−1.5399E−01	20
55	6.3333E+00	223	6.0000E+00*	1	6.6667E+00	9
56	−1.0000E+00	93	−1.0000E+00	2,075	−3.4560E+00	45
60	3.2570E−02	236	5.4709E−02*	465	3.2571E−02	29
61	−1.4365E+02	196	−1.4300E+02	621	0.0000E+00*	1
62	−2.5698E+04	33	−2.6272E+04	233	−2.6273E+04	32
63	9.6172E+02	159	9.6261E+02*	317	9.6172E+02	10
68	−9.2041E−01	439	−8.4354E−01*	1,316	−9.2042E−01	106
69	−9.5671E+02	581	−9.5665E+02*	2,471	−9.5107E+02	73
71	1.7014E+01	398	1.7031E+01*	1939	1.6000E+01*	1
73	2.9894E+01	305	3.0160E+01	223	2.9894E+01	16
74	5.1265E+03	279	5.1447E+03*	46,145	0.0000E+00*	1
75	5.1744E+03	2453	5.2331E+03*	22,678	0.0000E+00*	1
77	2.4151E−01	598	4.6807E+00*	1,904	2.4151E−01	85
78	−2.9197E+00	368	−2.8917E+00*	869	−2.9197E+00	27
79	7.8777E−02	495	2.4186E−01*	1,054	7.8777E−02	40
80	5.3950E−02	458	1.0000E+00*	557	5.3950E−02	23
81	5.3950E−02	481	9.9999E−01*	557	5.3950E−02	23
87	8.9276E+03	493	9.3254E+03*	16,244	4.2090E+04*	1
99	−8.3108E+08	777	−7.4573E+08*	729	−8.3108E+08	82

Table 2 Continued

Prob.	*Skinny*		HOPSPACK		DFO	
	f	#FE	f	#FE	f	#FE
107	5.0550E+03	858	5.0628E+03*	7,232	5.0550E+03	23
109	5.3621E+03	775	5.5010E+03*	57,551	0.0000E+00*	1
111	−4.7761E+01	4,607	–	–	−4.7760E+01	561
112	−4.7761E+01	1,815	−4.7761E+01	730	−4.7760E+01	85
114	−1.7688E+03	1,359	–	–	−1.7688E+03	276
119	2.4490E+02	1,331	2.4493E+02	944	2.4490E+02	90

The function values marked with '*' are infeasible solutions

numerical results, where the first column is the problem's number and, for each algorithm, f is the functional value and #FE is the number of objective function evaluations. A maximum of 1 hour of CPU time was allowed for each problem and '−' indicates that the algorithm has exceeded the maximum CPU time without returning information about the last iterate. The symbol '*' indicates that a feasible point could not be obtained.

Table 2 shows that *Skinny* was always able to find a feasible point. DFO failed to find feasible points in 6 problems because it started with an infeasible point and was unable to find two feasible points to construct the linear model. HOPSPACK failed to find feasible points in 22 problems.

The comparisons between *Skinny* and HOPSPACK show the advantages of separating the minimization from the feasibility process when the latter is computationally easier. The data profile in Fig. 1a shows that *Skinny* converged in 45 problems while HOPSPACK converged only in 16. Moreover, using the same number of function evaluations, *Skinny* solved approximately twice the number of problems that HOPSPACK solved. The number of infeasible points found by HOPSPACK can be decreased if we strengthen its stopping criteria, at the cost of an enormous increase in the number of function evaluations.

The comparison of *Skinny* and DFO in Fig. 1b shows that the first one solves 4 more problems. On the other hand, DFO is more efficient in the sense that, when both algorithms solve a problem, DFO spends less functional evaluations. Both features were expected since both the constraint-relaxing strategy and the poll step of *Skinny* aim to improve robustness, and may be unnecessarily conservative when the problems are easy.

In order to show how Δ controls the local optimality achieved by *Skinny*, we try to minimize $f(x) = x^2 \cos(x/3)$ subject to $x \in [-30, 30]$. This problem has 5 minimizers: $0, \pm 3\pi$, and $\pm 9\pi$, the last two being global minimizers. Using $x^0 = 0$ as starting point, with $\Delta = 1$ *Skinny* finds the point $x^* = 0$. If $\Delta = 10$ is used, the point $x^* = 3\pi$ is found and if $\Delta = 30$ is used, then the global minimizer $x^* = 9\pi$ is found. In practice, we also have to increase m_k when increasing Δ, since the number of iterations is finite and a sufficient number of poll directions have to be used in order to fulfill Assumption A2.

4.1 The case $w^k = 0$

The good performance of the stand-alone DFO and the fact that the convergence theory of *Skinny* does not depend on $w^k > 0$ raised the question as to the possible advantages of working with the original domains ($w^k = 0$) instead of w^k-feasible sets.

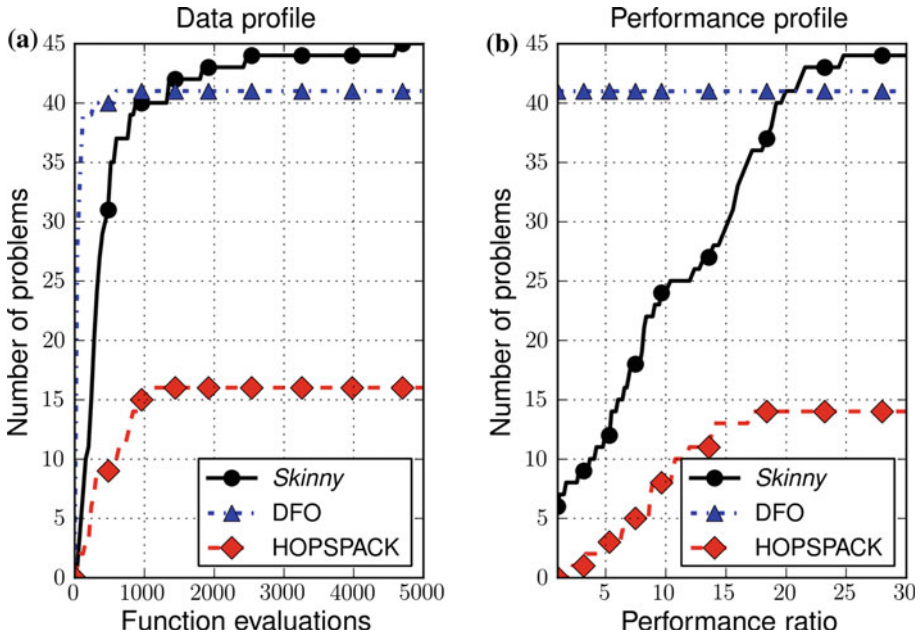

Fig. 1 Data (**a**) and performance (**b**) profiles for the comparison among *Skinny*, DFO and HOPSPACK

Table 3 *Skinny* with $w^k = 0$ performed better than *Skinny* using w^k-feasible sets and had a behavior similar to DFO

	Feas.	Conv.	Eff.	Function evaluations		
				$[0, 10^2]$	$(10^2, 10^3]$	$(10^3, 10^4]$
DFO	41	41	34	38/41	3/41	0/41
Skinny $w^k = 0$	47	46	13	42/47	3/47	2/47
Skinny	47	45	1	8/47	34/47	5/47

We compared the "$w^k = 0$" version of *Skinny* against its standard version and against DFO, in the same set of 47 problems described in Table 1. For each algorithm in Table 3, Feas represents the number of problems for which the solver has found a feasible solution, Conv represents the number of problems where there was convergence to a point with the best functional value, in the sense of (12), and Eff represents the number of problems for which the solver was the fastest. In the last three columns we show the number of problems in which each algorithm has spent up to 10^2, between 10^2 and 10^3 and more than 10^3 function evaluations, respectively. We can see that *Skinny* with $w^k = 0$ performs very similarly to DFO in terms of function evaluations and also inherits the good convergence properties of the theory presented here, converging in 46 problems.

The naive conclusion could be that it is better to use $w^k = 0$ instead of $w^k > 0$. However, the reason for the observed behavior is that, in these tests, it is not harder to work with the true domains than with their relaxations.

To show the advantages of working with a strictly positive w^k, suppose now that the feasible domain of the problem is **thin** and also **disconnected**. As an example, we present the

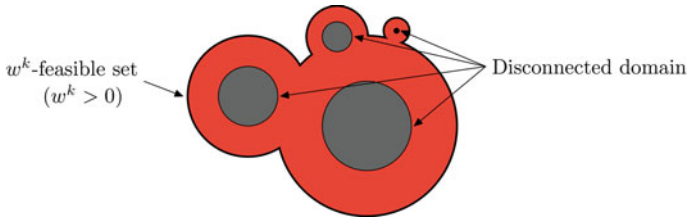

Fig. 2 A sufficiently positive value for w^k connects disconnected domains

integer programming problem of minimizing $f(x) = x^2$ subject to $x \in \{1, 2, 3, 4, 5\}$. The integrality constraint can be written as the differentiable equality constraint $\prod_{i=1}^{5}(x-i) = 0$ and the global minimizer is $x^* = 1$. Suppose that the starting point is $x^0 = 10$.

Using $w^k = 0$, *Skinny* first finds $y^1 = 5$ as a feasible point in the restoration step. There is nothing to be done by DFO, since this point is a local minimizer, and then, after the poll step, *Skinny* declares convergence to $\bar{x} = 5$. Now, using a sufficiently large w^k, all the disconnections in the domain disappear and the initial w^k-feasible set is an interval containing the points $\{1, 2, 3, 4, 5\}$. DFO is able to converge to the global unconstrained minimizer $x^1 = 0$ and, in the steps that follow, the global solution of the problem is found. Fig. 2 outlines the general idea of connection in disconnected domains when $w^k > 0$.

Another advantage of working with $w^k > 0$ is when a direct search method is more indicated for a problem than DFO. In the same way as DFO cannot deal with disconnected domains, direct search methods are not able to find feasible points on thin domains. As a consequence, if $w^k = 0$, *Skinny* would rely solely on the poll step, which could cause a decrease in the performance.

5 Final remarks

We presented a new algorithm for derivative-free constrained optimization which, under suitable assumptions, converges to local minimizers associated with a neighborhood of fixed size Δ. Convergence to global minimizers takes place if the feasible domain is bounded and Δ is large enough.

The new algorithm is based on successive minimizations on relaxed feasible sets. Convergence proofs follow even without feasibility relaxation, but, in this case, assumptions are harder to satisfy. We are especially interested in problems in which the main computational cost is associated with objective function evaluations, whereas the feasible set may exhibit high nonlinearity that could complicate the process of getting feasible points.

Handling thin feasible regions (and, in the limit, disconnected domains) may be particularly hard for algorithms based on direct search or Augmented Lagrangian approaches. Direct search methods have severe difficulties in preserving feasibility whereas penalty-like methods may spend many unnecessary functional evaluations motivated by the necessity of achieving feasibility.

We compared the new method against two well-established derivative-free algorithms: the first one is based on Augmented Lagrangians [25,33] and the second one is supported on trust-region ideas [11,12]. In principle, these algorithms are able to handle thin feasible regions. Moreover, the Augmented Lagrangian method may be implemented in such a way that convergence to global minimizers holds [8]. The numerical results show that our

algorithm spends less function evaluations than the Augmented Lagrangian method and always finds a feasible solution. Moreover, we showed the advantages of working with w^k-feasible sets ($w^k > 0$) in problems having disconnected thin domains. The new algorithm was shown to be able to solve more problems than the trust-region method DFO.

The flexibility of the new algorithm allows one to use different derivative-free methods both in the restoration phase and in the minimization phase.

The source code of the algorithm and the complete tables with the numerical results are available in www.ime.unicamp.br/~ra079963/skinny.

Acknowledgments We are indebt to Philippe Toint, who called our attention to the paper by Paviani and Himmelblau. We are also grateful to the two anonymous referees, whose comments helped to improve the quality of the present work, and Professor Susan Pyne for corrections of English.

References

1. Andreani, R., Birgin, E.G., Martínez, J.M., Schuverdt, M.L.: On augmented Lagrangian methods with general lower-level constraints. SIAM J. Optim. **18**, 1286–1309 (2007)
2. Anonymous: A new algorithm for optimization. Math. Program. **1**, 124–128 (1972)
3. Audet, C., Dennis, J.E. Jr.: Mesh adaptive direct search algorithms for constrained optimization. SIAM J. Optim. **17**(1), 188–217 (2006)
4. Audet, C., Dennis, J.E. Jr.: A progressive barrier for derivative-free nonlinear programming. SIAM J. Optim. **20**(1), 445–472 (2009)
5. Audet, C., Dennis, J.E. Jr., Le Digabel, S.: Globalization strategies for Mesh Adaptive Direct Search. Comput. Optim. Appl. **46**(2), 193–215 (2010)
6. Banerjia, C., Dwyer, R.A.: Generating random points in a ball. Commun. Stat. Simul. Comput. **22**(4), 1205–1209 (1993)
7. Bielschowsky, R.H., Gomes, F.A.M.: Dynamic control of infeasibility in equality constrained optimization. SIAM J. Optim. **19**(3), 1299–1325 (2008)
8. Birgin, E.G., Floudas, C.A., Martínez, J.M.: Global minimization using an augmented Lagrangian method with variable lower-level constraints. Math. Program. **125**(1), 139–162 (2010)
9. Birgin, E.G., Martínez, J.M.: Local convergence of an inexact-restoration method and numerical experiments. J. Optim. Theory. Appl. **127**, 229–247 (2005)
10. Conn, A.R., Gould, N.I.M., Toint, Ph.L.: Lancelot: A FORTRAN package for large-scale nonlinear optimization (Release A). Springer Publishing Company, Incorporated (2010)
11. Conn, A.R., Scheinberg, K., Toint, Ph.L.: Recent progress in unconstrained nonlinear optimization without derivatives. Math. Program. **79**(3), 397–414 (1997)
12. Conn, A. R., Scheinberg, K., and Toint, Ph. L.: A derivative free optimization algorithm in practice. In: Proceedings of 7th AIAA/USAF/NASA/ISSMO Symposium on Multidisciplinary Analysis and Optimization, St. Louis, MO (1998)
13. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization, MPS-SIAM Series on Optimization. SIAM, Philadelphia (2009)
14. Custódio, A.L., Vicente, L.N.: Using sampling and simplex derivatives in pattern search methods. SIAM J. Optim. **18**(2), 537–555 (2007)
15. Diniz-Ehrhardt, M.A., Martínez, J.M., Pedroso, L.G.: Derivative-free methods for nonlinear programming with general lower-level constraints. Comput. Appl. Math. **30**, 19–52 (2011)
16. Dolan, E.D., Moré, J.J.: Benchmarking optimization software with performance profiles. Math. Program. Ser. A **91**(2), 201–213 (2002)
17. Gould, N.I.M., Toint, Ph.L.: Nonlinear programming without a penalty function or a filter. Math. Program. **122**(1), 155–196 (2010)
18. Griffin, J.D., Kolda, T.G.: Nonlinearly-constrained optimization using heuristic penalty methods and asynchronous parallel generating set search. Appl. Math. Res. Express. AMRX **2010**(1), 36–62 (2010)
19. Hock, W., Schittkowski, K.: Test examples for nonlinear programming codes. Springer-Verlag New York, Inc., Secaucus, NJ, USA (1981)
20. Kolda, T.G., Lewis, R.M., Torczon, V.: Stationarity results for generating set search for linearly constrained optimization. SIAM J. Optim. **17**(4), 943–968 (2006)

21. Lewis, R.M., Torczon, V.: Pattern search algorithms for bound constrained minimization. SIAM J. Optim. **9**(4), 1082–1099 (1999)
22. Lewis, R.M., Torczon, V.: Pattern search algorithms for linearly constrained minimization. SIAM J. Optim. **10**(3), 917–941 (2000)
23. Lewis, R.M., Torczon, V.: A globally convergent augmented Lagrangian pattern search algorithm for optimization with general constraints and simple bounds. SIAM J. Optim. **12**(4), 1075–1089 (2002)
24. Le Digabel, S.: Algorithm 909: NOMAD: Nonlinear optimization with the MADS algorithm. ACM Trans.Math.Softw. 37(4), 44:1–44:15 (2011)
25. Lewis, R. M., Torczon, V.: A direct search approach to nonlinear programming problems using an augmented Lagrangian method with explicit treatment of linear constraints. Technical Report WM-CS-2010-01, College of William & Mary, Department of Computer Science (2010)
26. Martínez, J.M.: Inexact restoration method with Lagrangian tangent decrease and new merit function for nonlinear programming. J. Optim. Theory. Appl. **111**, 39–58 (2001)
27. Martínez, J.M., Pilotta, E.A.: Inexact restoration algorithms for constrained optimization. J. Optim. Theory. Appl. **104**, 135–163 (2000)
28. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. SIAM J. Optim. **20**(1), 172–191 (2009)
29. Muller, M.E.: A note on a method for generating points uniformly on N-dimensional spheres. Commun. ACM **2**, 19–20 (1959)
30. Nelder, J.A., Mead, R.: A simplex method for function minimization. Comput. J. **7**, 308–313 (1965)
31. Paviani, D.A., Himmelblau, D.M.: Constrained nonlinear optimization by heuristic programming. Oper. Res. **17**, 872–882 (1969)
32. Pedroso, L.G.: Programação não linear sem derivadas. PhD thesis, State University of Campinas, Brazil, (2009). (in Portuguese)
33. Plantenga, T.D.: HOPSPACK 2.0 User Manual. Sandia National Laboratories, Albuquerque, NM and Livermore, CA, SAND2009-6265 (2009)
34. Powell, M.J.D.: The BOBYQA algorithm for bound constrained optimization without derivatives. Cambridge NA Report NA2009/06, University of Cambridge, Cambridge (2009)
35. Schrage, L.: A more portable Fortran random number generator. ACM Trans. Math. Softw. **5**(2), 132–139 (1979)
36. Torczon, V.: On the convergence of pattern search algorithms. SIAM J. Optim. **7**(1), 1–25 (1997)
37. Vicente, L.N., Custódio, A.L.: Analysis of direct searches for discontinuous functions. Math. Program. **133**(1–2), 299–325 (2012)