submitted manuscript No.
(will be inserted by the editor)

Unified approach for solving Box-Constrained models with continuous or discrete variables by Non monotonous Derivative Free Optimization techniques.

Ubaldo M. García-Palomares
Pedro S. Rodríguez-Hernández

Received: date / Accepted: date

Abstract This paper describes a unified approach for solving Box-Constrained Optimization Problems (BCOP) in Euclidian spaces. The variables may be either continuous or discrete; in which case, they range on a grid of isolated points regularly spaced. For the continuous case, convergence is shown under standard assumptions; for the discrete case, slight modifications ensure that the algorithm stops in a finite time. Moreover, function evaluations are carried out only on feasible points on the grid, avoiding spurious computations on non feasible points. The paper describes a pseudo code and a preliminary code is written in C, which is applied to small models that have been suggested in the open literature.

Keywords DFO · discrete variable · non monotone

1 Introduction

We deal with the problem (G) of minimizing a function $f(.)$ defined on a grid $\mathcal{G} \subset \mathbb{R}^n$ enclosed in a Box;

$$\textbf{Problem G:} \qquad \min f(x),\ x \in \mathcal{E} = \{x \in \mathcal{G} : s \leq x \leq t\}, \tag{1}$$

where $s \leq x \leq t$ should be understood componentwise, that is, $s^k \leq x^k \leq t^k, k = 1, \ldots, n$. With no loss of generality we can assume that $s^k < t^k, k = 1, \ldots, n$ because all those variables subjected to $s^k = x^k = t^k$ for some $1 \leq k \leq n$ may be deleted from the model. Formally, the Grid-set $\mathcal{G}$ is defined as:

$$(x \in \mathcal{G}) \Leftrightarrow (x + Gz \in \mathcal{G}),\ \text{for any vector } z \in \mathbb{Z}^n \text{ of integer values}, \tag{2}$$

Ubaldo M. García Palomares Pedro S. Rodríguez-Hernández
Universidade de Vigo, Information Technology Group, Atlantic
Tel.: +34 986813464 Tel.; +34 986812174
Fax: +34 986812116
E-mail: ubaldo@gti.uvigo.es E-mail: pedro@gti.uvigo.es

where $G \in R^{nxn}$ is a diagonal matrix with components $G^{kk} \triangleq g^k > 0, k = 1, \ldots, n$, which represent the grid size on the canonical axis. Figure 1 depicts a set $\mathcal{G} \subset \mathbb{R}^2$, with $g^1 = 2, g^2 = 3$, i.e., $G = \begin{bmatrix} 2 & 0 \\ 0 & 3 \end{bmatrix}$.

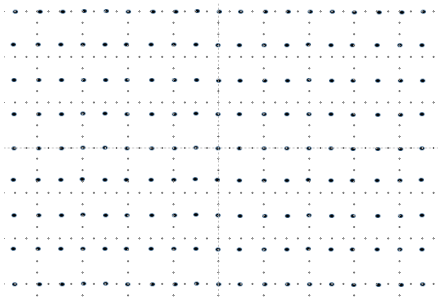

Fig. 1 Grid-set $\mathcal{G} \subset R^2, g^1 = 2, g^2 = 3$

There is a vast class of techniques, denoted as derivative free optimization (DFO), that have been developed to solve optimization models for which derivatives are non existent or non reliable. These technique include, among others, direct search search methods (DSM), surrogate models and stochastic methods. We develop a DSM for the solution of box constrained optimization problems with either discrete or continuous variables. We must be aware of *a*) the plethora of algorithms that use pure or hybrid DFO techniques for solving Problem R with continuous variables formulated as:

$$\textbf{Problem R:} \qquad \min f(x), x \in \mathcal{F} = \{x \in \mathbb{R}^n : s \leq x \leq t\}, \tag{3}$$

and *b*) the numerical results that have been reported for this problem (See for instance [4,7,8,10,11] and references therein). A collection of numerical examples that includes test problems from [4] can be downloaded from [12].

Due to the inherent difficulties that arise for problems with discrete variables, special techniques have been proposed for solving **Problem G** (1). See [9] and references therein. Nonetheless, we are proposing algorithms that solve **Problem R** and **Problem G** using the same DFO technique, which is a slight variation of the algorithm described by García et al [4]. The *new* algorithm solves **Problem G** in a finite processing time. At the same time, the algorithm avoids any computation of function values on any $x \notin \mathcal{G}$.

We will be using a rather standard notation with the following peculiarities:

•$\mathbb{R}_+, \mathbb{Z}_+$, are, respectively, the set of non negative scalars and the set of non negative integers,

•Superindices represent components of a vector (matrix), generally defined in $\mathbb{R}^n (R^{nxn})$, and subindices represent different vectors,

•$D = \{d_1, \ldots, d_q\}$ is a set of q bounded directions,

•$D^+ = \left\{ \sum_{k=1}^{q} \alpha_k d_k : \alpha_k \geq 0, d_k \in D \right\}$ is the *non negative* span of D.

•$\sigma(\tau) : \mathbb{R}_+ \to \mathbb{R}_+$ is strictly positive, i.e., $(\tau > 0) \Rightarrow (\sigma(\tau) > 0)$, and $\lim_{\tau \to 0} \sigma(\tau)/\tau = 0$,

•If $u, v \in \mathbb{R}^n, uv^T$ is the nxn U matrix, with $U^{jk} = u^j v^k$, and $u^T v$ is the inner product $\sum_{k=1}^{n} u^k v^k$

•The subindex i stands for iteration, although it could be dropped when it seems clear fom the context. $(x_i, \tau_i, \varphi_i, D_i)_{i \in K} \to (\bar{x}, \bar{\tau}, \bar{\varphi}, \bar{D})$ means that there exists $K \subseteq \mathbb{N}$ such that $(\bar{x}, \bar{\tau}, \bar{\varphi}, \bar{D})$ is an accumulation point of the (sub)sequence $(x_i, \tau_i, \varphi_i, D_i)_{i \in K}$.

The rest of the paper is organized as follows: In Section 2 we propose a framework for solving both Problem G and Problem R. The algorithm for solving the latter problem is akin to the one suggested in [4] and is endowed with the same convergence properties. Minor modifications give rise to a new algorithm tailored for solving Problem G. This Section highlights the minor differences between both algorithms. In particular, for Problem G, a set of search directions is suggested that ensures finite termination under adequate conditions. Moreover, it impels the set of points generated to remain in $\mathcal{E}$. Section 3 is devoted to implementation issues that enhance the algorithms' performance. Preliminary numerical results on small problems are also given.

2 Framework for solving Problems G and R

We will describe our algorithms by pseudo codes intermingled with line numbers to ease the presentation.

2.1 Solving Problem R

Figure 2 describes the pseudocode of the algorithm that generates a (sub)sequence converging to a stationary point of (3).

Theoretically the following assumptions are required:

A1. The set $\{x \in \mathcal{F} : f(x) \leq \varphi_0\}$ is compact

A2. $f(\cdot)$ is Lipschitzian.

A3. $\{D_i\}_{i \in \mathbb{N}} \to \bar{D}$ and $\bar{D}^+ = \mathbb{R}^n$. Also, e_k and $-e_k$ must belong to D_i whenever x_i^k is ϱ-quasi active, that is, either $x_i^k < s^k + \varrho$ or $x_i^k > t^k - \varrho$.

$\varrho > 0$ allows the algorithm to conform the search directions to the geometry of the model. In real applications, assumptions **A1** and **A2** must be taken for granted, that is, no practitioner should attempt to verify its certainty. However, careful implementation is required to ensure **A3**. Since the sets $\{D_i\}_{i \in \mathbb{N}}$ play a key role in our algorithms we briefly describe the construction displayed in Fig. 3. The search directions defined by (4) are the columns of the orthogonal Householder matrix $H = (I - 2uu^T/u^T u)$, with $u^k = 0$ if x^k is ϱ-quasi active as indicated in Fig. 3. Note that $(u_i^k = 0) \Rightarrow (H_i e_k = e_k)$, and **A3** holds.

Since the algorithm described in Figure 2 *slightly* differs from the one proposed in [4] it is convenient to prove some lemmas and sketch the convergence theorem.

1. **Input:** $x_0 \in \mathcal{F},\ f_0 \leq \varphi_0,\ 0 < \epsilon < \tau_0 \in \mathbb{R}_+,\ MaxIter$
2. $i \leftarrow 0; \quad iter \leftarrow MaxIter$
3. WHILE $(\tau_i > \epsilon)$
4. Generate $D_i = \{d_{i1}, \ldots, d_{iq}\}$ by Fig. 3
5. IF $(\exists d \in D_i) : (x_i + \tau_i d) \in \mathcal{E}$, and $f(x_i + \tau_i d) < \varphi_i - \sigma(\tau_i)$
6. Choose $x_{i+1} \in \mathcal{E} : f_{i+1} < \varphi_i - \sigma(\tau_i)$
7. $\tau_{i+1} \leftarrow \tau_i; \quad iter \leftarrow iter - 1$
8. IF (iter $= 0$)
9. $\varphi_{i+1} \leftarrow \varphi_i - \sigma(\tau_i); \quad iter \leftarrow MaxIter$
10. ELSE
12. $\varphi_{i+1} \leftarrow (1-\beta) f_{i+1} + \beta \varphi_i$ for some $\beta \in [0\ 1]$
13. ENDIFELSE
14. ELSE
15. $\tau_{i+1} \leftarrow \mu \tau_i$ for some $\mu \in [0.3\ 0.7]$
16. $x_{i+1} \leftarrow x_i;\ \varphi_{i+1} \leftarrow \varphi_i$
17. ENDIFELSE
18. $i \leftarrow i + 1$
19. ENDWHILE
20. **Output:** x_i

Fig. 2 Basic algorithm for solving Problem R (3)

Let $\varrho^k > 0, k = 1, \ldots, n$;
Let $Q_i = \{k : x_i^k < s^k + \varrho^k \text{ or } x_i^k > t^k - \varrho^k\}$ be the index set of all ϱ-quasi active variables;
let $u_i \in \{u \in \mathbb{R}^n : \left\langle \begin{array}{l} u^k = 0, \text{for } k \in Q_i \\ u^k \in [-1\ 1], k \notin Q_i \end{array} \right\rangle$, or $0 = u_i \in \mathbb{R}^n$.
let $H_i = I$, if $u_i = 0$; *and*
$H_i = I - 2u_i u_i^T / u_i^T u_i$, otherwise.

$$\text{Define } D_i = \{\pm H_i e_1, \ldots, \pm H_i e_n\} \tag{4}$$

Fig. 3 Construction of the set of search directions for Problem R

Lemma 1 *$\{\varphi_i\}_{i \in \mathbb{N}}$ is a non increasing sequence. Besides, $f_i \leq \varphi_i$ for all i.*

Proof φ_i is modified in lines 9, 12 and 16. In all these instances $\varphi_{i+1} \leq \varphi_i$ and the first part of the lemma is valid. To prove that $f_i \leq \varphi_i$ for all $i \in \mathbb{N}$, we note that line 1 states that $f_0 \leq \varphi_0$; lines 12 and 6 state that $\varphi_{i+1} = \varphi_i - \sigma(\tau_i) > f_{i+1}$, and lines 12 and 6 state that $\varphi_{i+1} = (1-\beta) f_{i+1} + \beta \varphi_i > f_{i+1}$. The proof follows by induction. □

Lemma 2 $\{\tau_i\}_{i \in \mathbb{N}} \to 0$

Proof Lines 7 and 15 make sure $\{\tau_i\}_{i \in \mathbb{N}}$ is non increasing and bounded below; therefore it has a limit. We assume that $\{\tau_i\}_{i \in \mathbb{N}} \to \bar{\tau} > 0$ and will exhibit a contradiction. If this is the case, $\{\sigma(\tau_i)\}_{i \in \mathbb{N}} \geq \varepsilon > 0$ by definition. Besides, line 15 will be executed finitely many times and from some iteration $\imath$ onwards the algorithm will execute an infinite loop between lines 6 and 12. Let $i_1, i_2, \ldots$ be those iterations for which line 8 holds. We obtain for any i_j that

$$\varphi_{i_j+1} = \varphi_{i_j} - \sigma(\tau_{i_j}) \leq \varphi_{i_{j-1}+1} - \sigma(\tau_{i_j}),$$

and consequently $\varphi_{i_j+1} \leq \varphi_{i_{j-1}+1} - \varepsilon$ occurs infinitely often and $\{\varphi_i\}_{i\in\mathbb{N}}$ decreases without bounds, but since $\varphi_i \geq f_i, i \in \mathbb{N}$ we deduce that $f(\cdot)$ does not have a minimum in a compact set. A contradiction. □

The convergence result of the algorithm described in Figure 2 is summarized in the following proposition:

Proposition 1 *Under assumptions* **A1 - A3**, *there exists* $K \subseteq \mathbb{N}$ *such that* $\{x_i, \tau_i, D_i\}_{i\in K} \to (\bar{x}, 0, \bar{D})$ *and:*

– *if* $\lim_{\{x_i\}\to\bar{x},\tau_i\downarrow 0} \dfrac{f(x_i+\tau_i d)-f(x_i)}{\tau} = \nabla(\bar{x})^T d$ *for any* $d \in \mathbb{R}^n$

then $\nabla f^k(\bar{x}) \begin{cases} \geq 0, \text{ if } \bar{x}^k = s^k, \\ \leq 0, \text{ if } \bar{x}^k = t^k, \\ = 0, \text{ otherwise} \end{cases}$

– *if*

$$\begin{array}{c} f(x+\tau d) - f(x) = \tau f'(x,d) + o(\tau), \text{ and} \\ f'(x,\alpha d) = \alpha f'(x,d) \text{ for any } \alpha > 0, d \in \mathbb{R}^n \end{array} \tag{5}$$

then $f'(\bar{x}, d) \geq 0$ *for any feasible direction* $d \in \bar{D}$

Two important consequences can be derived from (5):

1. $f'(x,d) = \lim_{\tau\to 0} \dfrac{f(x+\tau d)-f(x)}{\tau}$, and
2. $f'(x,d) \geq 0$ for all feasible d is a necessary condition for x being a minimizer. See references [4,6].

Note that for non smooth functions the convergence theorem only ensures that the directional derivative $f'(x,d)$ is non negative at feasible directions $d \in D$. It is therefore convenient to enlarge D with more search directions. It is now common to ask for the set D to be asymptotically dense in a ball surrounding x.

2.2 Algorithm for solving Problem G

Before proceeding with the algorithm it is convenient to define a stationary point for Problem G. First, we define the *local neighborhood*

$$V(\bar{x}) = \{x \in \mathcal{E} : x = \bar{x} \pm g^k e_k, k = 1, \ldots, n\}, \tag{6}$$

and now we define a stationary point $\bar{x} \in \mathcal{E}$ as

Definition 1 $\bar{x} \in \mathcal{E}$ is stationary for Problem P if

$$(x \in V(\bar{x})) \Rightarrow (f(x) \geq f(\bar{x})) \tag{7}$$

1. **Input:** $x_0 \in \mathcal{E},\ f_0 \leq \varphi_0,\ 0 < \zeta_0 \in \mathbb{Z}_+,\ MaxIter$
2. $i \leftarrow 0;\quad iter \leftarrow MaxIter$
3. WHILE $(\zeta_i > 0)$
4. Generate $D_i = \{d_{i1}, \ldots, d_{iq}\}$ by Fig. 5
5. IF $(\exists d \in D_i) : (x_i + d) \in \mathcal{E}$, and $f(x_i + d) < \varphi_i$
6. Choose $x_{i+1} \in \mathcal{E} : f_{i+1} < \varphi_i$
7. $\zeta_{i+1} \leftarrow \zeta_i;\quad iter \leftarrow iter - 1$
8. IF $(iter = 0)$
9. $\varphi_{i+1} \leftarrow f_{i+1};\quad iter \leftarrow MaxIter$
10. ELSE
12. $\varphi_{i+1} \leftarrow (1 - \beta) f_{i+1} + \beta \varphi_i$ for some $\beta \in [0\ 1]$
13. ENDIFELSE
14. ELSE
15. $\zeta_{i+1} \leftarrow (\zeta_i - 1) - \lfloor \mu \zeta_i \rfloor$ for some $\mu \in [0\ 0.3]$
16. $x_{i+1} \leftarrow x_i;\ \varphi_{i+1} \leftarrow \varphi_i$
17. ENDIFELSE
18. $i \leftarrow i + 1$
19. ENDWHILE
20. **Output:** x_i

Fig. 4 Basic algorithm for solving Problem G (1)

Figure 4 describes the algorithm pseudo code for solving (1) intermingled with line numbers that we will frequently refer to in the sequel. This new algorithm shares some features with its sibling described by Figure 2; in particular, it is possible to apply the algorithm described by Figure 4 to a discretized version of problem R (3) and still obtain good results. This approach was suggested by García et al [3].

Our first task is to construct an appropriate $\{D_i\}_{i \in \mathbb{N}}$ satisfying assumption **A3**, but with one additional ingredient. It should ensure that

$\langle x \in \mathcal{G}, d \in D_i \rangle \Rightarrow x + d \in \mathcal{G}$

Let $\varrho^k > g^k, k = 1, \ldots, n$;
Let $Q_i = \{k : x_i^k \leq s^k + \varrho^k$ or $x_i^k \geq t^k - \varrho^k\}$ be the index set of all g-quasi active variables,
let $\zeta > 0$,

let $u_i \in \{u \in \mathbb{R}^n : \left\langle \begin{array}{l} u^k = 0, \text{for } k \in Q_i \\ u^k \in \{-1,\ 0,\ 1\} \\ u^T u = \zeta \end{array} \right\rangle$, or $0 = u_i \in \mathbb{R}^n$.

let $H_i = I$, if $u_i = 0$, and
$H_i = (u_i^T u_i) I - 2 u_i u_i^T$, otherwise.

$$\text{Define } D_i = \{\pm G H_i e_1, \ldots, \pm G H_i e_n\} \tag{8}$$

Fig. 5 Construction of the set of search Directions for Problem G

Several comments are in order:

– We have chosen $\varrho^k = 2g^k$ to ensemble the index set Q_i of g-quasi active variables. A bigger number might force $u_i = 0$, which in turn forces the

search on the coordinate axis; but this has not been, in general, the best choice when solving Problem R (3).
- $H_i e_k = (u_i^T u_i)e_k - 2u^k u_i$, for $k = 1, \ldots, n$ are column vectors of integer components; which implies $x_i \pm GH_i e_k \in \mathcal{G}, k = 1, \ldots, n$, as desired.
- We do not pick the search directions in an ordered fashion $d_1, d_2, d_3, \ldots, d_q$. They are randomly taken from the set.
- u^k is randomly chosen from 3 possible values; $u^k \in \{-1, 0, 1\}$. Other choices are possible. We decided to satisfy $u^T u = \xi$ to facilitate the proof of Proposition 2.
- Note that $u \neq 0 \Rightarrow ||H_i e_k|| = (u^T u) = \zeta, k = 1, \ldots, n$. If in addition, $G = I$, which happens when we deal with integers, the function values are computed on points located on the ball $B(x_i) = \{x \in \mathbb{R}^n : ||x - x_i|| = \zeta\}$. If in the continuous version τ happens to be an integer, then its search coincides with the discrete version.
- As H_i is orthogonal and G is a diagonal matrix with positive values, the matrix GH_i is nonsingular and the set of search directions defined by (8) positively spans $\mathbb{R}^n$ as required by **A3**.

Lemma 3 *$\{\varphi_i\}_{i \in \mathbb{N}}$ is a non increasing sequence. Besides, $f_i \leq \varphi_i$ for all i.*

Proof In line 16 $\varphi_{i+1} = \varphi_i$. Line 6 states that $f_{i+1} < \varphi_i$; therefore in line 9 $\varphi_{i+1} = f_{i+1} < \varphi_i$, and in line 12 $\varphi_{i+1} = (1 - \beta)f_{i+1} + \beta\varphi_i < \varphi_i$; and the first part of the lemma is valid. To prove that $f_i \leq \varphi_i$ for all $i \in \mathbb{N}$, we note that line 1 states that $f_0 \leq \varphi_0$. Besides, we have that either $f_{i+1} = \varphi_{i+1}$ in line 9 or $\varphi_{i+1} = (1 - \beta)f_{i+1} + \beta\varphi_i > f_{i+1}$ from lines 12 and 6. The proof follows by induction. □

Theorem 1 *The algorithm stops at some iteration i.*

Proof We assume that the theorem does not hold and exhibit a contradiction. If the theorem is false, line 15 of the algorithm is not executed from some $\bar{\imath}$ onwards; that is, the IF clause from line 6 to 9 is always executed. Let $i_1, i_2, \ldots$ be those iterations for which line 8 holds. We then have by construction and the previous lemma that $f_{i_j+1} = \varphi_{i_j+1} < \varphi_{i_j} \leq \varphi_{i_{j-1}+1} = f_{i_{j-1}+1}$. Consequently, $f_{i_j+1} < f_{i_{j-1}+1}$ occurs infinitely often, but this is impossible because the set $\mathcal{E}$ is finite by assumption **A1**. □

Corollary 1 *For some* finite $i, \zeta_i = 0$.

Proof It is implied by the previous theorem. □

We are now ready to show that the output x_{i+1} given in Figure 4 satisfies some necessary condition. More specifically, next proposition shows that x_{i+1} satisfies the definition 1.

Proposition 2 *The output x_{i+1} of algorithm G satisfies (7).*

Proof By the previous corollary and the updating of ζ in line 15 we must have $\zeta_i = 1$ for some i. By construction $u_i^T u_i = 1$; which means that only one component of u is not null, say $u^k = \pm 1, u^j = 0, j \neq k$. Hence,

$He_k = (I - 2u_i u_i^T)e_k = -e_k$, and $He_j = e_j, j \neq k$.

The algorithm searches on $V(x_i)$; therefore $\zeta_i = 0$ if (and only if) (7) holds. □

3 Numerical issues

The algorithms described in Fig. 2 and Fig. 4 work nicely, but our implementation forces $s^k \geq -50$ and $t^k \leq 50, k = 1, \ldots, n$. Besides, the algorithms' performance is enhanced with typical strategies that have been used in previous works. We now enumerate some of these strategies. A C code with all these features is a companion of this paper that can be downloaded from [2].

3.1 Implementation notes

1. **The value of $MaxIter$.**
   The IF clause in line 8, is reminiscent of the strategy used in [5] in an algorithm for *global* minimization of unconstrained minimization problems with continuous variables. Any strategy that makes sure that φ sufficiently decreases for Problem R (or strictly decreases for Problem G) every finite number of iterations suffices. Numerical tests, not reported here, reveal that our algorithms' performance is not too sensible to this finite number. In our implementation we used $MaxIter = 20$.
2. **The choice of $\sigma(\tau)$.**
   In our proof we need $\sigma(\cdot)$ to be strictly positive. For numerical reasons we chose: $\sigma(\tau) = 0.01$ IF $\tau > 0.1$, and
   $\sigma(\tau) = 10\tau^2(1 + |\varphi|)$ otherwise.
3. **The choice of φ.**
   We recall that $\{f_i\}_{i \in \mathbb{N}} \leq \varphi_0$; hence the *hill climbing* capacity of the algorithm will be limited if we choose $\varphi_0 = f_0$. On the other hand, local search is enforced. If the practitioner strives for a global minimum, (s)he might pick a large value for φ_0, but this, in general, increases the number of function evaluations.
4. **The stopping criterion.**
   It is nowadays desirable to stop the algorithm once the budget has been depleted. This is in general estimated by the number of function evaluations. It is therefore advisable to stop the algorithms once the maximum number of function evaluations allowed is surpassed.
5. **Increase of τ and ζ.**
   τ and ζ may increase without preventing the convergence of the algorithms. In our implementation we chose to increase these values when lines 6 to 12 are executed on more than one half of the search directions, in which case $\tau \leftarrow \gamma\tau$; $\zeta \leftarrow \lfloor \gamma\zeta \rfloor$ with γ picked from a uniform distribution in [1 2].

6. **The direction generator u**.
   We chose $u^k = 0, k \in Q$, as required in Figures 3 and 5.
   For Problem R, $u^k, k \notin Q$ is picked from a uniform distribution in [-1 1].
   For Problem G, $u^k, k \notin Q$ is picked randomly from {-1, 0, 1}.
   There exist many possibilities to construct u, as long as $u \in \mathbb{R}^n$ for Problem R, and $u \in \mathbb{Z}^n$ for Problem G. The choice suggested for problem R has been successful in numerical tests [4]. This is however, the first attempt with Problem G and more evidence is needed.
7. **Choice of search directions**.
   We could include as many search directions as desired in D, as long as (4, 8) are preserved. In our implementation we did not augment D. Once the set $D = \{d_1, d_2, \ldots, d_q\}$ is determined, all search directions are selected at random.
8. **Extrapolation on search direction**.
   When a direction $d \in D$ has been identified in line 5, the algorithms try to get $\eta > 1$ and $y = (x + \eta d)$ satisfying both conditions on line 5.
9. **The output x_{i+1}**.
   Due to the non monotonous feature of our algorithms, it could happen that x_{i+1} is not the real solution. The algorithm keeps track of the best value found (x_b, f_b), which becomes the returned variable.
10. **Additional search on random starting points**.
    Before outputting (x_b, f_b) the algorithms generate a random feasible point and repeat the process. If no improvement is achieved after a certain number of attempts, the algorithm stops. This strategy was suggested by Addis and Locatelli [1], but we should keep in mind that the number of function evaluations might increase significantly.

3.2 Numerical tests

The algorithms were coded in *C*, compiled with the *gcc* compiler version 4.8 and run on a 64 bit desk computer equipped with a 3.30GHz Intel processor under windows 10. We wrote a POSIX compliant code that can be compiled and run under any operating system. At all cases we imposed a budget limit of 80000 function evaluations. We focus our tests on the discrete case, because it is the main contribution of this paper.

We attempt to solve the continuous problem (3) on a grid in order to evaluate the discretization of the objective function as a practical procedure. A discrete version of (3) may be obtained by replacing x by $\bar{x}+Gz, \bar{x} \in \mathcal{F}, z \in \mathbb{Z}^n$. We ran our tests with $G = gI$, for different grid size g. We are reporting here our results with 2 well known functions: Rosenbrok and Shekel. Both of these functions have an all-integer solution vector; therefore algorithm R (Fig. 2) and algorithm G (Fig. 4) applied to a discretization of problems (9) and (10) should yield the same result, as long as the solution remains in $\mathcal{G}$. Another common feature of both functions is that they have local minima with *almost* integer values. The 10-variable Rosenbrok function has a local integer

minimizer x_d =(-1 1...1), and its continuous counterpart has a solution x_c : $||x_d - x_c||_\infty < 0.2$. Shekel has 10 all-integer minimizers; indeed, each column of the matrix C is a minimizer, and the global one is $\bar{x}$= (4 4 4 4), with $f(\bar{x}) = -10.53$. We ran two kind of experiments: starting with a local minimizer and random start. The starting point for Rosenbrok was x_0 = (-1 1...1), and Shekel was started on the local minimizer farthest away from the global minimizer, namely x = (2 9 2 9). It is worth mentioning that both algorithms escaped from the local starting minimizer. However, implementation note 9 makes sure the algorithms will *recall* a *visit* to the global minimizer.

10-variable Rosenbrok (n= 10):

$$\begin{aligned} \underset{x\in\mathbb{R}^n}{\text{minimize}} \ & \sum_{k=1}^{n-1} \left[10(x^{k+1} - x^k \ x^k)\right]^2 + \left[1 - x^k\right]^2 \\ \text{subject to:} \quad & -5 \le x^k \le 5, k = 1, \ldots, n, \end{aligned} \tag{9}$$

4-variable Shekel:

$$\begin{aligned} \underset{x\in\mathbb{R}^4}{\text{minimize}} \ & -\sum_{j=1}^{10} \left[\sum_{k=1}^{4} \left[x^k - C^{kj}\right]^2 + b^j\right]^{-1} \\ \text{subject to:} \quad & 0 \le x^k \le 10, \ k = 1, 2, 3, 4, \end{aligned} \tag{10}$$

$$\text{b= } \tfrac{1}{10} \ (1 \ 2 \ 2 \ 4 \ 4 \ 6 \ 3 \ 7 \ 5 \ 5),$$

$$C = \begin{bmatrix} 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \\ 4 & 1 & 8 & 6 & 3 & 2 & 5 & 8 & 6 & 7 \\ 4 & 1 & 8 & 6 & 7 & 9 & 3 & 1 & 2 & 3 \end{bmatrix}$$

Other results on small problems are not reported here, but seem to confirm that discretization may be a viable strategy, but *scaling* of the function might be needed. Steep functions like Rosenbrok's may be difficult because its values at grid points tend to be not only high but also close to each other, which may trigger the number of function evaluations. The results gathered in Table 1 also shows in the column *success* the number, in percentage, that the global minimum was found.

Final remark: From the results shown in Table 1 it is worth mentioning that a better *success* with a significant reduction in function evaluations *may* be obtained with the discretization of the continuous problem. This is a promising result and we plan to carry out experiments on medium to large dimensional realistic models. It also seems that this work can lead us to the solution of models with mixed variables with a unified approach using variable decomposition.

Acknowledgements This work was partially supported by "Ministerio de Economía y Competitividad" (project COINS TEC2013-47016-C2-1-R) and Xunta de Galicia, grant GRC2014/046, Spain.

Table 1 Numerical tests (1000 runs per case)

Rosenbrok:	Starting at local minimizer		Random start	
Grid size g	Function Evals min - avg - max	Success %	Function Evals min - avg - max	Success %
1	4805 - 6263 - 7899	100	5478 - 7734 - 13528	99.9
0.5	6907 - 9875 - 17966	96.4	7820 - 13726 - 31195	44.0
0.2	10157 - 15510 - 29611	8.1	11505 - 22114 - 50958	10.7
0.1	13081 - 20955 - 36106	3.5	13982 - 25216 - 53798	7.1
0.001	80015 - 80015 - 80015	0.1	91352 - 159959 - !!!	2.8
continuous	63254 - 132002 - 193675	2.5	112616 - 113446 - 114276	0.2

Shekel:	Starting at local minimizer		Random start	
Grid size g	Function Evals min - avg - max	Success %	Function Evals min - avg - max	Success %
1	445 - 1071 - 2733	42.6	533 - 1160 - 2734	41.6
0.5	747 - 1507 - 3455	84.2	774 - 1417 - 3831	89.1
0.2	1149 - 2008 - 3770	11.6	1100 - 1577 - 4297	26.1
0.1	1408 - 2381 - 3431	1.3	1499 - 1794 - 3291	20.3
0.001	???	0	4187 - 4735 - 5223	23.6
continuous	14346 - 37195 - !!!	42	7481 - 36475 - 72897	34

??? The global minimum was not reported.
!!! Budget depleted

References

1. Bernardetta Addis, Marco Locatelli. A new class of test functions for global optimization. *Journal of Global Optimization*, 38: 479-501, (2007)
2. Ubaldo M. García-Palomares. Software Companion for DFO approach for bounded and discrete variables. *Researchgate* DOI 10.13140/RG.2.2.27405.13284, (2016)
3. Ubaldo M. García-Palomares, Enrique Costa-Montenegro, Rafael Asorey-Cacheda, Francisco J. González-Castaño. Adapting derivative free optimization methods to engineering. models with discrete variables. *Optimization and Engineering*, 13(4): 579-594, (2012)
4. Ubaldo M. García-Palomares, Ildemaro J. García-Urrea, Pedro S. Rodríguez-Hernández. On sequential and parallel non monotone derivative free algorithms for Box Constrained optimization. *Optimization Methods and Software*, 28(6): 1233-1261, (2013)
5. Ubaldo M. García-Palomares, Francisco J. González-Castaño, Juan C. Burguillo-Rial. A combined global & local search (CGLS) approach to global optimization. *Journal of Global Optimization*, 34: 409-426, (2006)
6. Ubaldo M. García-Palomares, José Félix Rodríguez. New sequential and parallel derivative-free algorithms for unconstrained minimization. *SIAM Journal On Optimization*, 13(1): 79-96, (2002)
7. Serge Gratton, Philipe L. Toint, Anke Troltzsch. An active-set trust-region method for derivative-free nonlinear bound-constrained optimization. *Optimization Methods and Software*, 26(4): 873-894, (2011)
8. Robert M. Lewis, Virginia Torczon. Pattern search algorithms for bound constrained minimization. *SIAM Journal on Optimization*, 9: 1082-1099, (1999).
9. Kien-Ming Ng. A continuation approach for solving nonlinear optimization problems with discrete variables. *Dissertation Thesis, Stanford University*, (2002)
10. Michael J.D. Powell. The BOBYQA algorithm for bound constrained optimization without derivatives. *Cambridge Report NA2009/06, University of Cambridge*, (2009)

11. Luis M. Ríos, Nikolaos V. Sahidinis. Derivative-free optimization: a review of algorithms and comparison of software implementations. *Journal of Global Optimization*, 56: 1247-1293, (2013)
12. Pedro S. Rodríguez-Hernández. http://webs.uvigo.es/pedro.rodriguez/pub/DFO_GGR, (2016)