# GOSH: derivative-free global optimization using multi-dimensional space-filling curves

Daniela Lera* and Yaroslav D. Sergeyev†

**Abstract**

Global optimization is a field of mathematical programming dealing with finding global (absolute) minima of multi-dimensional multiextremal functions. Problems of this kind where the objective function is non-differentiable, satisfies the Lipschitz condition with an unknown Lipschitz constant, and is given as a "black-box" are very often encountered in engineering optimization applications. Due to the presence of multiple local minima and the absence of differentiability, traditional optimization techniques using gradients and working with problems having only one minimum cannot be applied in this case. These real-life applied problems are attacked here by employing one of the mostly abstract mathematical objects – space-filling curves. A practical derivative-free deterministic method reducing the dimensionality of the problem by using space-filling curves and working simultaneously with all possible estimates of Lipschitz and Hölder constants is proposed. A smart adaptive balancing of local and global information collected during the search is performed at each iteration. Conditions ensuring convergence of the new method to the global minima are established. Results of numerical experiments on 1000 randomly generated test functions show a clear superiority of the new method w.r.t. the popular method DIRECT and other competitors.

**Key Words**. Global optimization, space-filling curves, derivative-free methods, acceleration, Lipschitz functions.

# 1 Introduction

Many real-world optimization problems are stated as a global optimization problem since functions describing these applications are often multiextremal, non-differentiable, and hard to evaluate even at one point (see, for example [17, 21, 22, 31, 34,

*Dipartimento di Matematica e Informatica, Università di Cagliari, Cagliari, Italy

†Corresponding author; Dipartimento di Ingegneria Informatica, Modellistica, Elettronica e Sistemistica, Università della Calabria and the Institute of High Performance Computing and Networking of the National Research Council of Italy, Via Pietro Bucci 42C, 87036 Rende (CS), Italy; and Department of Software and Supercomputing, Lobachevskiy University of Nizhni Novgorod, Gagarin Av. 23, Nizhni Novgorod, Russia

35, 47, 52]). In this paper, we focus our attention on continuous global optimization problems

$$\min\{F(y) : \ y \in S = [a, b]\}, \tag{1.1}$$

where $S$ is a hyperinterval in $\mathbf{R}^N$ and the objective function $F(y)$ can be multiextremal, non-differentiable, and given as a "black-box", i.e., any information regarding its analytical representation or any other data describing its structure is not available. However, it is supposed that $F(y)$ satisfies the Lipschitz condition

$$|F(y') - F(y'')| \leq L\|y' - y''\|, \qquad y', y'' \in S, \tag{1.2}$$

with an unknown Lipschitz constant $L$, $0 < L < \infty$, in the Euclidean norm. This statement can be very often encountered in practice and in the literature there exist numerous methods for dealing with the problem (1.1), (1.2) (see, e.g., [1, 3, 4, 5, 13, 17, 21, 23, 32, 33, 34, 41, 47, 48, 51, 52]).

In this paper, we consider the applied problem (1.1), (1.2) by using one of the mostly abstract mathematical objects – space-filling curves introduced by Peano in 1890 and independently by Hilbert in 1891 (even though we use Hilbert's version of the curves, the traditional terminology for this kind of objects is "Peano curves" due to the priority of Peano). The curves under consideration emerge as the limit objects generated by an iterative process. They are fractals constructed using the principle of self-similarity. It is possible to prove that the curves fill in the hypercube $S \subset \mathbf{R}^N$, i.e., they pass through every point of $S$ (this fact gave rise to the term "space-filling curves"). It is known that it is possible to reduce the dimension of the global optimization problem (1.1), (1.2) by using the curves and to move from a multivariate problem to a univariate one (see studies in this direction in [2, 38, 44, 45, 47, 46]).

More precisely, it can be shown (see [2, 45, 47]) that, by using space-filling curves, the multi-dimensional global minimization problem (1.1), (1.2) can be turned into a one-dimensional problem and that finding the global minimum of the Lipschitz function $F(y), y \in S \subset R^N$, is equivalent to determining the global minimum of the one-dimensional function $f(x)$ over the interval $[0, 1]$, i.e., it follows

$$f(x) = F(p(x)), \qquad x \in [0, 1], \tag{1.3}$$

where $p(x)$ is the Peano curve. Moreover, the Hölder condition

$$|f(x') - f(x'')| \leq H|x' - x''|^{1/N}, \qquad x', x'' \in [0, 1], \tag{1.4}$$

holds (see [47]) for the function $f(x)$ with the constant

$$H = 2L\sqrt{N + 3}, \tag{1.5}$$

where $L$ is the Lipschitz constant of the original multi-dimensional function $F(y)$ from (1.1), (1.2). In Fig. 1-right, the reduced function in one dimension corresponding to the test function in two dimensions from Fig. 1-left is shown. Clearly, a

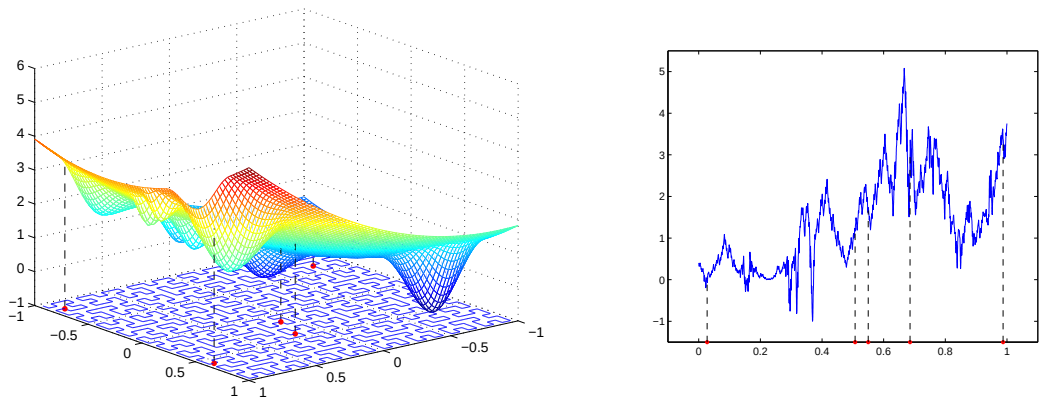

Figure 1: *A two-dimensional function from [10] satisfying the Lipschitz condition together with an approximation of level 5 to Peano curve (left) and the corresponding univariate Hölderian function (right). Red dots show points on the curve where the objective function has been evaluated.*

numerical approximation of the Peano curve is used in computations for the reduction. Thus, one can try to attack the problem (1.1), (1.2) by proposing algorithms for minimizing Hölderian function (1.3), (1.4) in one dimension.

It can be seen from the statement of the original problem (1.1), (1.2) that the only available information regarding the multi-dimensional function $F(y)$ is that $F(y)$ satisfies the Lipschitz condition (1.2) with an unknown constant $L$. As a result, the way the Lipschitz information is used by an optimization algorithm becomes crucial for its performance, convergence, and speed. In the literature there exist several methods to estimate $L$ (see [4, 11, 12, 15, 16, 17, 18, 42, 43, 44, 47, 50]), and it is known that an overestimation of $L$ may slow down the search whereas an underestimate of the constant can lead to loss of the global solution. Let us briefly describe methods used to estimate $L$.

First, there exist algorithms that for the whole domain $S$ use the same a priori given estimate of $L$ or its adaptive estimate recalculated during the search at each iteration (see, e.g., [4, 17, 18, 33, 34, 36, 43, 44, 47]). This approach does not take into account any local information about the behavior of the objective function over small subregions of the domain $S$. This drawback can slow down the search significantly. A more advanced approach proposed originally in [39, 40] suggests to adaptively approximate local Lipschitz constants $\tilde{L}(D_j)$ in different subregions $D_j \subset S$ of the search region $S$ during the process of optimization. This procedure performs a local tuning on the behavior of the objective function balancing global and local information obtained during the search (see also interesting hybridization ideas in [49, 50]). It has been shown in [20, 24, 39, 44, 47] that the local tuning techniques can lead to a significant acceleration of the global search. Another interesting approach that has been introduced in [19] in the popular method called DIRECT uses at each iteration several estimates of the Lipschitz constant $L$ simultaneously. This way

to deal with Lipschitz information attracts a wide interest of researchers (see, e.g., [6, 7, 8, 9, 19, 23, 29, 30, 31, 32, 33]) and is under scrutiny in this work, as well.

In this paper, we propose to use Peano curves and instead of using the Lipschitz information in many dimensions to work with the Hölder information in one dimension trying to obtain several estimates of the Hölder constant using the DIRECT methodology. It should be stressed that such a transposition of the approach is not trivial at all. In fact, in the literature (see [14, 24, 25, 27, 28, 44]) there exist several methods estimating global and local Hölder constants whereas the usage of the DIRECT approach encounters a number of serious difficulties (see [26]) in the context of Hölder optimization. In Section 2, we describe a strategy that solves them and allows us to work with several estimates of the Hölder constant at each iteration. Then, a two-phases procedure intended to accelerate the search is presented in Section 3. A new algorithm using both discoveries for solving the problem (1.1), (1.2) and its convergence properties are described in Section 4. Section 5 presents results of numerical experiments that compare the new method with its competitors on 1000 test functions randomly generated by the GKLS-generator from [10]. Finally, Section 6 contains a brief conclusion.

# 2 Two ways to represent Hölderian minorants

Due to the use of the Peano space-filling curves, the $N$-dimensional problem (1.1), (1.2) is turned into the one-dimensional problem (1.3), (1.4) with the one-dimensional objective function $f(x)$ from (1.3) satisfying the Hölder condition (1.4) with a constant $0 < H < \infty$ over the interval $[0, 1]$. It follows from (1.4) that, for all $x, z \in [0, 1]$ we have

$$f(x) \geq f(z) - H|x - z|^{1/N}. \tag{2.1}$$

This fact means that the function

$$G(x) = f(z) - H|x - z|^{1/N},$$

with $z \in [0, 1]$ fixed, is a minorant (or support function) for $f(x)$ over $[0, 1]$, i.e.

$$f(x) \geq G(x), \qquad x \in [0, 1].$$

Analogously, if we consider subintervals $d_i = [a_i, b_i]$, $1 \leq i \leq k$, belonging to $[0, 1]$ we obtain that the following function

$$G^k(x) = g_i(x), \;\; x \in [a_i, b_i], \;\; 1 \leq i \leq k, \tag{2.2}$$

$$g_i(x) = \begin{cases} g_i^-(x) = f(m_i) - H(m_i - x)^{1/N}, & x \in [a_i, m_i], \\ g_i^+(x) = f(m_i) - H(x - m_i)^{1/N}, & x \in [m_i, b_i], \end{cases} \tag{2.3}$$

$$m_i = (a_i + b_i)/2 \tag{2.4}$$

is a discontinuous nonlinear minorant for $f(x)$ (see Fig. 2) and the values $R_i$, $1 \leq i \leq k$, are lower bounds for the function $f(x)$ over each interval $d_i$, $1 \leq i \leq k$. These

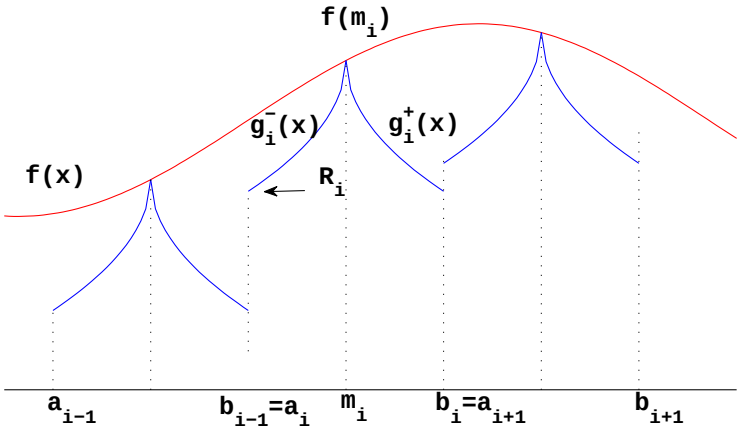


Figure 2: *Hölder support functions.*

values are called *characteristics of intervals* and can be calculated as follows if an overestimate $H_1 \geq H$ of the Hölder constant $H$ is given

$$R_i = R_i(H_1) = \min_{x \in [a_i, b_i]} g_i(x) = f(m_i) - H_1|(b_i - a_i)/2|^{1/N}. \tag{2.5}$$

As was mentioned in the introduction, the DIRECT algorithm (see [19]) uses at each iteration several estimates of the Lipschitz constant for selecting a suitable set of subintervals in the central points of which to evaluate the objective function. This selection can be easily done thanks to a smart representation of the intervals in a diagram in two dimensions. This representation is the core point of DIRECT and can be done since the Lipschitz information is used by this method to produce piecewise linear minorants. In order to use the same methodology in the framework of the Hölderian optimization it is necessary to be able to find a suitable representation of intervals, as well.

Let us try to do this following the idea of DIRECT and show that a simple transposition from Lipschitz to Hölder world does not work. We represent in a two-dimensional diagram each interval $d_i = [a_i, b_i]$ by a point with coordinates $(h_i, f(m_i))$, where $h_i = 0.5(b_i - a_i)$ and $m_i$ is from (2.4) exactly as DIRECT does. In Fig. 3-left, we have represented five different intervals $d_A$, $d_B$, $d_C$, $d_D$, and $d_E$ by the points $A$, $B$, $C$, $D$, and $E$, respectively. If we consider a fixed overestimate $H_1$ of the Hölder constant, we can observe the corresponding nonlinear support functions (2.3) (shown in blue solid lines) related to these intervals. The characteristic $R_A(H_1)$

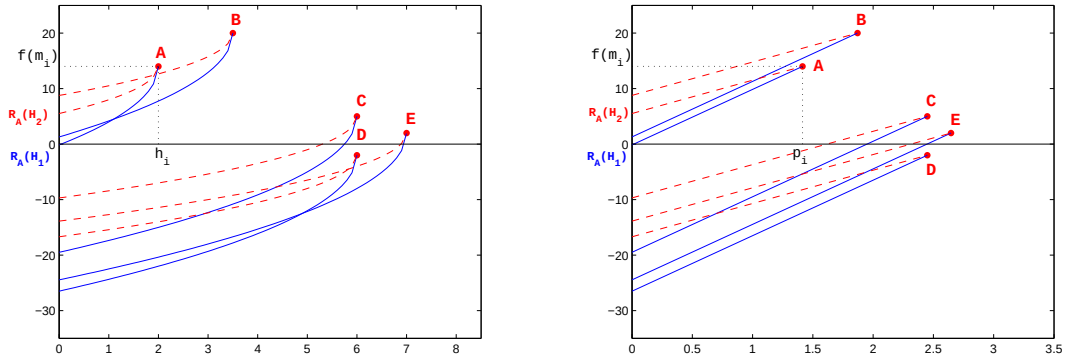


Figure 3: *Representation of intervals in the Euclidean metric (left) and in the Hölderian metric (right).*

of the interval represented by the dot $A$ is obtained as the intersection of the curve (2.3) constructed at the point $A$ with the vertical coordinate axis. It can be seen that the best (the lowest) characteristic is $R_D(H_1)$ and the interval $d_D$ would be subdivided at the next iteration if $H_1$ is chosen as the estimate for $H$. However, the choice of $R_D(H_1)$ is not easy since, as it can be seen from Fig. 3-left, the curves constructed using the estimate $H_1$ intersect one another in various ways.

In addition, remind that we do not know the real value of $H$ and wish to try all possible estimates of $H$ from zero to infinity. The auxiliary functions corresponding to the second estimate $H_2$ are shown in Fig. 3-left by red dashed lines. They produce again a lot of intersections among themselves and with the curves corresponding to $H_1$. It becomes clear that the selection of the lowest characteristic for all possible estimates of $H$ even with such a small number of intervals becomes complicated and it is unclear how to select intervals by varying estimates of the Hölder constant from 0 to infinity.

In order to overcome this difficulty and to give a more transparent procedure for selection of the best characteristics, a different representation of the intervals is proposed. The idea consists of the usage of the metric of Hölder instead of the Euclidean one in the construction of the diagram. More precisely, a generic interval $d_i = [a_i, b_i]$ belonging to a current partition $\{D^k\}$ at the $k$th iteration is represented by a dot $P_i$ with the coordinates $(p_i, w_i)$ where

$$p_i = |(b_i - a_i)/2|^{1/N}, \qquad w_i = f(m_i), \tag{2.6}$$

and $m_i$ is from (2.4).

In Fig. 3-right, the representation of the same five intervals considered in Fig. 3-left can be observed in the new metric. A great simplification can be clearly seen since there are no more nonlinear curves and intersections between them for each fixed estimate of $H$. The obtained diagram is very similar to that used by the DIRECT method, in the Lipschitzian case [19]. In Fig. 3-right, the characteristic $R_A(H_1)$ of the interval represented by the point $A$ is exactly the intersection of

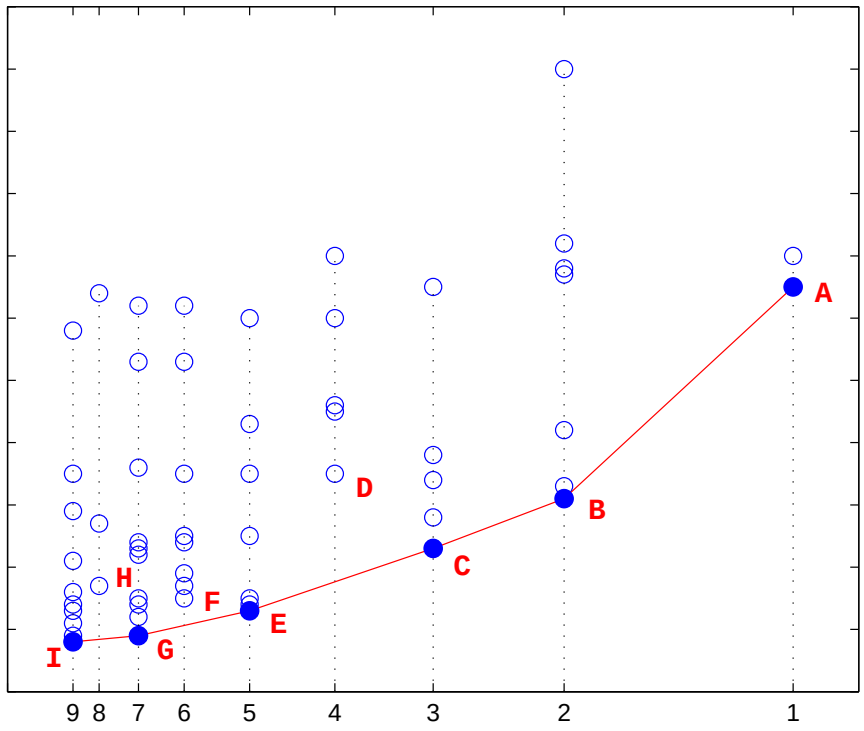


Figure 4: *The nondominated intervals $d_A$, $d_B$, $d_C$, $d_E$, $d_G$ and $d_I$ are represented by dots $A$, $B$, $C$, $E$, $G$ and $I$.*

the line passing through $A$ with slope $H_1$ and the vertical coordinate axis. Notice that, as expected, the values in the vertical coordinate axis coincide with those of Fig. 3-left. The selection of intervals with the best characteristics corresponding to different estimates of $H$ becomes so much easier and is discussed in the following two sections.

# 3 Selection of intervals: Two-phase approach

In this section, we describe in detail the intervals selection procedure that will be used in the method to be introduced in Section 4. As was already said above, at each iteration $k$ the method should select in a suitable way a promising set of subintervals in which it intends to intensify the search and execute new trials (*trial* is evaluation of $f(x)$ at a point $x$ that is called *trial point*). To accelerate the search, a two-phase technique that balances the global and local information collected during the work of the method is introduced.

In order to describe the selection procedure let us discuss Fig. 4 that shows a possible scenario at a generic iteration $k$ of the algorithm. The interval $[0, 1]$

(remind that since Peano curves are applied, the search is performed over the one-dimensional interval $[0,1]$ (see (1.3))) is subdivided into subintervals $d_i = [a_i, b_i]$, $i = 1, ..., I(k)$, belonging to the current partition $D^k$. Each interval is represented by a point in the two-dimensional diagram in Fig. 4, with coordinates given by (2.6), and is characterized, for each fixed value of $H$, by a lower bound given by $R_i$ from (2.5). Points with the same abscissa represent intervals that have the same width. In Fig. 4, there are nine different groups of intervals corresponding to the points $A$, $B$, ..., $I$. At each iteration $k \geq 1$ of the method each group of intervals receives a positive integer index $l = l(k)$. The first group of large intervals (the column of the dot $A$ in Fig. 4) gets the index $l = 1$, and the subsequent intervals are identified progressively by indices 2, 3, 4, ... etc. So, in Fig. 4 there are nine groups with indices 1, 2, ..., 9. The index 9 is referred to the group of intervals with minimal width (column of the point $I$).

For any fixed value $H$ of the Hölder constant, it is easy (see Fig. 3-right where lower bounds for $H = H_1$ and $H = H_2$ are shown) to identify the interval corresponding to the minimal lower bound with respect to the other intervals in the current partition. By varying the value of $H$ from 0 to infinity, the method should select a set of intervals corresponding to the smallest lower bound from (2.5) for some particular estimate of the Hölder constant $H$. These intervals should be partitioned during the next iteration and are called *nondominated* intervals and it can be easily seen that they are located on the lower-convex hull of the set of dots representing the intervals. In Fig. 4 the nondominated intervals are identified by points located at the bottom of each group with the same horizontal coordinate, that is points $A$, $B$, $C$, $E$, $G$ and $I$. In practice, to determine these intervals algorithms for identifying the convex hull of the dots can be used, for example, the algorithm called Jarvis march, or gift wrapping, see [37]. Notice that the points $H$, $F$, and $D$ do not represent nondominated intervals even though they are the lowest in their groups. This happens because (see, e.g., the point $F$) the point $G$ dominates $F$ at smaller values of Hölder constant $H$ and the point $E$ dominates $F$ at higher values of $H$.

The two phases (that can interchange each other several times during the work of the method) are the following: investigation of large unexplored intervals in order to find attraction regions of local minimizers that are better than the current best found solution (global phase) and a local improvement of the current best found solution (local phase). In order to explain their functioning let us remind that all the intervals on the diagram (see Fig. 4) are ordered in the increasing order from smaller to larger intervals along the horizontal axis. Thus, well explored zones of the search region corresponding to attraction regions of already visited local minima are located on the left-hand part of the diagram (small intervals) whereas unexplored zones of the domain are represented on the right-hand part of the diagram (large intervals). If during the work of the global phase a better solution than the current one has been obtained, then the method switches to the local phase in order to improve the new best record. After several improving steps the method switches back to the global phase and the search of new promising minima continues until

the satisfaction of a stopping rule.

During the global phase the new algorithm explores mainly large intervals, thus it identifies the set of nondominated intervals not among all groups of intervals but only among some groups with indices lower than a calculated "middle index" $r$. This index represents a separator between the groups of large intervals and small ones. The global phase is performed until a function value improving the current minimal value on at least 1% is obtained. When this happens, the method switches to the local phase in the course of which the obtained new solution is improved locally. In the case when the algorithm is not switched to the local phase during more than a fixed number $IglobMax$ of iterations (the improvement of the current minimum is still not found by exploring large intervals), it performs one "security" iteration in which determines nondominated intervals considering all groups of intervals present in the diagram.

Thus, during each iteration of the global phase the algorithm identifies a set of nondominated intervals. The subdivision of each of these intervals is performed only if a significant improvement on the function values with respect to the current minimal value $f_{min}(k)$ is expected, i.e., once an interval $d_t \in \{D^k\}$ becomes nondominated, it can be subdivided only if the following condition is satisfied

$$R_t(\tilde{H}) \leq f_{min}(k) - \xi, \tag{3.1}$$

where the lower bound $R_t = R_t(\tilde{H})$ is from (2.5) and the parameter $\xi$ prevents the algorithm from subdividing already well-explored small subintervals.

During the local phase improving the just found new best solution the algorithm always explores three intervals: the interval containing the best current point (best interval) and the intervals located on the right and on the left of it. This phase finishes when the width of at least one of these intervals is less than a given accuracy. After the end of the local phase the algorithm switches back to the global phase and tries to find better solutions that can be located far away from the current best point. Notice that during the local phase a security iteration is carried out after performing a fixed number $IlocMax$ of iterations without switching to the global phase. This is done in order to avoid a too long concentrating of efforts at local minima that are not global solutions. As before, at the security iteration nondominated intervals among all groups of intervals present in the diagram are taken into consideration.

Once the selection phase (local or global) has been concluded, the chosen intervals are subdivided in order to produce new trial points by the following partition strategy. At a generic iteration $k$, let $S_k$ be the set of the intervals to be partitioned and $d_t = [a_t, b_t]$ be an element of $S_k$ represented by the corresponding point in the diagram at Fig. 4. Each interval $d_t$ of the set $S_k$ is subdivided into three equal parts

$$[a_t, b_t] = [a_t, u_t] \cup [u_t, v_t] \cup [v_t, b_t], \tag{3.2}$$

of the length $(b_t - a_t)/3$, with

$$u_t = a_t + (b_t - a_t)/3, \quad v_t = b_t - (b_t - a_t)/3. \tag{3.3}$$

The three new generated intervals are added to the current partition $\{D^k\}$ and to the diagram in Fig. 4 and the interval $[a_t, b_t]$ is deleted from both. Finally, two new trials, $f(c_1)$ and $f(c_2)$, are executed at the central points of the new intervals $[a_t, u_t]$ and $[v_t, b_t]$, where

$$c_1 = (a_t + u_t)/2, \qquad c_2 = (v_t + b_t)/2. \tag{3.4}$$

Notice that the midpoint of the third interval $[u_t, v_t]$ is also the midpoint of the initial interval $[a_t, b_t]$ and, therefore, the function $f(x)$ has already been calculated in it at previous iterations.

We conclude this section by reminding that the objective function $f(x)$ is obtained by applying Peano curve that theoretically is introduced as a limit object being a fractal constructed using principles of the self-similarity. In practice, computable approximations of the Peano curve are used. Let us denote them by $p_M(x)$, where $M$ is the level of approximation of the curve (see the approximations with $M = 5$ in Fig. 1, respectively). The choice of the level $M$ of the curve is essential to obtain a good performance of the method: in fact, a level that is too low can be insufficient to fill in the domain in an appropriate way creating so a risk to lose the optimal solution. On the other hand, when the value of $M$ increases, the function in one dimension becomes more oscillating, especially if the dimension $N$ of the original problem (1.1) grows up (see [28] for a detailed discussion). With increasing the dimension $N$, the width of intervals selected for partitioning can become very small (remind that we are in $[0, 1]$ and the metric of Hölder is used) and even get close to the computer precision. For these reasons it is required an additional check of the width of the interval before subdivision. Namely, the interval $d_t = [a_t, b_t]$ is partitioned only if the following condition is satisfied

$$b_t - a_t > \delta, \tag{3.5}$$

where $\delta$ is a parameter of the method.

# 4 The GOSH algorithm

In this section, a new algorithm called *GOSH* (*G*lobal *O*ptimization algorithm working with a *S*et of estimates of the *H*ölder constant) is presented.

To describe the algorithm formally, we need to specify some notations. Suppose that at an iteration $k \geq 1$ a partition $\{D^k\}$ of $D = [0, 1]$ has been obtained. Suppose also that each interval $d_i \in \{D^k\}$ is represented by a dot in the two-dimensional diagram from Fig. 4 and each group of intervals with the same width is numbered by the same integer index: this index is an integer positive number that varies between $imax(k)$ (index that identifies the column of the larger intervals) and $imin(k)$ (index of the column of the smaller intervals). The following notations are also adopted:

$f_{min}(k)$ is the best function value (the "record" value) at the iteration $k$, and $x_{min}(k)$ is the corresponding coordinate.

$d_{min}(k)$ is the interval containing the point $x_{min}(k)$.

$f_{prec}(k)$ is the old best record. It serves to memorize the record $f_{min}(k)$ at the start of the current phase (local or global).

*Lcount* and *Gcount* are counters of iterations performed during the local and global phases, respectively.

$IlocMax$ and $IglobMax$ are maximal allowed numbers of iterations that can be executed during the local and global phases, respectively, before making the general security iteration (in which the nondominated intervals are selected from the entire search domain).

*phase* is a flag specifying the current phase. It is equal to "loc" and "glob" in the local and global phases, respectively.

$p_M(x)$ is the $M$-approximation of the Peano curve.

$S^k$ is the set of intervals, $S^k \subset D^k$, that will be subdivided and the corresponding set $J^k$ is the set of their indices.

*jloc* is a flag that takes into account the fact that the set $S^k$ can be empty. In this case $jloc = 0$, otherwise $jloc = 1$.

We are ready now to describe the algorithm.

**Algorithm GOSH**

**Step 0.** (Initialization). Set the current iteration number $k := 1$.

Split the initial interval $D = [0, 1]$ in three equal parts and set $x^1 = 1/6$, $x^2 = 1/2$, $x^3 = 5/6$ and compute the values of the function $z^j = f(x^j) = F(p_M(x^j))$, $j = 1, 2, 3$.

Set the current partition of the search interval $D^1 = \{[0, 1/3], [1/3, 2/3], [2/3, 1]\}$.

Set the current number of intervals $I = 3$ and the current number of trials $T = 3$.

Set $f_{min}(1) = \min\{z^1, z^2, z^3\}$, and $x_{min}(1) = \arg\min\{f(x^i) : i = 1, 2, 3\}$.

Set $phase = loc$, $Lcount = Gcount = 0$.

After executing $k$ iterations, the iteration $k+1$ consists of the following steps.

**Step 1.** (Intervals selection) Identify the set $S^k$, $S^k \subset D^k$, and the corresponding set $J^k$ as follows.

**Step 1.1** (Global phase) **if** $(phase == glob)$ **then**
**if** $(Gcount < IglobMax)$
Determine nondominated intervals that satisfy conditions (3.1) and (3.5) by considering only groups of intervals with indices

going from $imax(k)$ up to $r(k) = \lfloor (p(k) + imax(k))/2 \rfloor$, where $\lfloor x \rfloor$ denotes the integer part of $x$ and $p(k)$ is the index of the group the interval $d_{min}(k)$ belongs to.
$Gcount = Gcount + 1$

**elseif** $(Gcount == IglobMax)$
Determine nondominated intervals that satisfy conditions (3.1) and (3.5) by considering all the groups of intervals with indices between $imax(k)$ and $p(k)$
$Gcount = 0$

**endif**

**Step 1.2** (Local phase) **if** $(phase == loc)$ **then**
$jloc = 1$

**if** $(Lcount < IlocMax)$
Determine the interval $d_{min}(k)$ and the two intervals, denoted by $dr_{min}(k)$ and $dl_{min}(k)$ located on the right and on the left of it, respectively. They are selected only if the condition (3.5) is satisfied.
$Lcount = Lcount + 1$

**elseif** $(Lcount == IlocMax)$
Determine nondominated intervals that satisfy conditions (3.1) and (3.5) by considering all the groups of intervals with indices between $imax(k)$ and $p(k)$.
$Lcount = 0$

**endif**

**endif**

Include found intervals in the set $S^k$ and their indices in the set $J^k$.
If $S^k = \emptyset$ then $jloc = 0$ and go to Step 3.

**Step 2.** (Subdivision of intervals) Set $D^{k+1} = D^k$ and perform Steps 2.1–2.3.

**Step 2.1** (Interval selection). Select a new interval $d_t = [a_t, b_t]$ from $S^k$ such that

$$t = \arg\max_{j \in J^k} \{b_j - a_j\}.$$

**Step 2.2** (Subdivision and sampling). Subdivide interval $d_t$ in three new equal subintervals, named $d_{t1}$, $d_{t2}$, $d_{t3}$ of the length $(b_t - a_t)/3$ following (3.2), (3.3) and produce two new trial points accordingly to (3.4). Eliminate the interval $d_t$ from $D^{k+1}$, i.e., set $D^{k+1} = D^{k+1} \setminus \{d_t\}$, and update $D^{k+1}$ with the insertion of the three new intervals, i.e.,

$$D^{k+1} = D^{k+1} \cup \{d_{t1}\} \cup \{d_{t2}\} \cup \{d_{t3}\}.$$

Increase both the current number of intervals $I = I + 2$, and the current number of trials $T = T + 2$.

Update the current record $f_{min}$ and the current record point $x_{min}$, if necessary.
Set $amp(j) = (b_t - a_t)/3$, $j \in J^k$.

**Step 2.3** (Next interval). Eliminate the interval $d_t$ from $S^k$, i.e., set $S^k = S^k \setminus \{d_t\}$ and $J^k = J^k \setminus \{t\}$.

If $S^k \neq \emptyset$, then go to Step 2.1. Otherwise calculate $amploc = \min_{j \in J^k} amp(j)$ and go to Step 3.

**Step 3.** (Switch)
**if** $(f_{min}(k) \leq f_{prec}(k) - 0.01 \cdot |f_{prec}(k)|)$
$f_{prec}(k) = f_{min}(k)$
if $(phase == glob)$ then $Lcount = 0$ endif
$phase = loc$
**elseif** $(phase == loc \ .\&. \ amploc \geq \delta' \ .\&. \ jloc == 1)$
$phase = loc$
**else**
if $(phase == loc)$ then $Gcount = 0$ endif
$phase = glob$
**endif**

**Step 4.** (End of the current iteration). Increase the iteration counter $k = k + 1$. Go to Step 1 and start the next iteration.

Different stopping criteria can be used in the $GOSH$ algorithm introduced above. One of them will be introduced in the next section presenting numerical experiments.

Let us make some comments upon the introduced method. Step 1 is the phase of selection of the intervals that, as was said above, can be either global or local. Suppose that at a generic iteration $k$ of the algorithm the situation is that shown in Fig. 4, with 9 different groups of intervals, and assume that the interval $d_{min}(k)$ containing the current minimum point $x_{min}(k)$, belongs to the group of intervals identified by the index 7 (so exactly the point $G$). If $phase = loc$ then 3 intervals will be selected: $d_{min}(k)$, that corresponds to the point $G$ in the diagram Fig. 4 and the intervals located to the right and to the left of it in $[0, 1]$, respectively. Notice, that the latter two intervals, namely $dr_{min}(k)$ and $dl_{min}(k)$, can belong to two different groups of intervals in the diagram and not necessarily to the group with the index 7. In contrast, if the situation where $phase = glob$ takes place then the separator index $r$ is calculated where $r = \lfloor \frac{7+1}{2} \rfloor = 4$ and the nondominated intervals are searched only among the groups of intervals from index 1 to index 4. In this example, intervals represented by the points $A$, $B$, and $C$ at the diagram in Fig. 4 will be selected and split in three parts. Dots $A$, $B$, and $C$ will disappear from the diagram and there will be three new points in the column of $B$, three in the column of $C$, and three in that of $D$.

If in the local phase it happens that $Lcount = IlocMax$ (or, analogously, in the global phase $Gcount = IglobMax$) then nondominated intervals among all groups

of intervals are retrieved. Thus, in the diagram at Fig. 4 intervals represented by points $A$, $B$, $C$, $E$, $G$, and $I$ will be split. The three intervals obtained by the interval $d_I$ will be represented by three points in the newly created column with the index 10. Notice that only intervals that satisfy condition (3.5) are selected for the further subdivision. It should be also emphasized that in Step 3, at the situation $phase = loc$, the local exploration continues until the width of at least one of the 3 selected intervals is smaller than a fixed $\delta' \geq \delta$, with $\delta$ from (3.5).

Let us consider now convergence properties of the $GOSH$ algorithm. The first result discusses a connection between the original multi-dimensional problem and the reduced univariate one. To obtain the latter problem and to go to the interval $[0, 1]$, an approximation $p_M(x)$ of the Peano curve of a fixed level $M$ is applied and in the course of the algorithm a lower bound $U^*_M$ of the multi-dimensional function $F(y)$ is calculated along the curve. In order to return to the original problem (1.1), (1.2) in $N$ dimensions, it is important to understand how a lower bound for $F(y)$ over the entire domain $[a, b]$ in $\mathbf{R}^N$ can be obtained from $U^*_M$. The following theorem gives the answer to this problem.

**Theorem 4.1** *Let $U^*_M$ be a lower bound along the space-filling curve $p_M(x)$ for a multi-dimensional function $F(y)$, $y \in [a, b] \subset R^N$, satisfying Lipschitz condition with constant $L$, i.e.,*

$$U^*_M \leq F(p_M(x)), \qquad x \in [0, 1].$$

*Then the value*

$$U^* = U^*_M - 2^{-(M+1)} L \sqrt{N}$$

*is a lower bound for $F(y)$ over the entire region $[a, b]$.*

**Proof.** See [28] or the recent monograph [44] for the proof of this result. □

Theorem 4.1 is important because it links the multi-dimensional problem (1.1), (1.2) to the one-dimensional problem (1.3), (1.4), so we can concentrate our attention on the convergence properties in the one-dimensional interval $[0, 1]$. Let us suppose that the maximal number of generated trial points tends to infinity, and prove that the infinite sequence of trial points generated by the $GOSH$ converges to any point of the one-dimensional search domain. This kind of convergence is called *everywhere dense* convergence.

**Theorem 4.2** *If $\delta = 0$ in (3.5), then for any point $x \in [0, 1]$ and any $\eta > 0$ there exists an iteration number $k(\eta) \geq 1$ and a trial point $x^{i(k)}$, $k > k(\eta)$, such that $|x - x^{i(k)}| < \eta$.*

**Proof.** In the selection Step 2 of the algorithm the two phases, local and global, are alternated. In the local phase of $GOSH$ an interval is subdivided only if its width is greater than a fixed $\delta' > 0$, $\delta'$ from Step 3 of $GOSH$. When the width of the selected interval becomes less than $\delta'$, the algorithm switches to the global

phase. Since it is assumed that $\delta = 0$ in (3.5), and since the one-dimensional search region has a finite length and $\delta'$ is a positive finite number, then there exists a finite iteration number $j = j(\delta')$ such that, for all iterations greater than $j$, only the global phase will be used during the work of the $GOSH$.

In the global phase the algorithm $GOSH$ always selects for partitioning at least one interval $d_t$ from the group of largest intervals (in Fig. 4 the group with index 1). In fact, there always exists a sufficiently large estimate $H_\infty$ of the Hölder constant $H$, such that the interval $d_t$ is the nondominated interval with respect to $H_\infty$, and condition (3.5) is satisfied. Therefore, at each iteration, the intervals with the largest width will be partitioned into three subintervals of the length equal to a third of the length of the subdivided interval. Notice that each group of intervals contains only a finite number of intervals since the interval is finite and all its subintervals have a finite length. Thus, after a sufficiently large number of iterations $k > k(\eta)$, all the intervals of the group with the maximal width will be partitioned. Such a procedure will be repeated with a new group of the largest intervals (the group with index 2 in Fig. 4) and so on until the largest intervals of the current partition will have the length smaller than $\eta$. As a result, in the neighborhood of radius $\eta$ of any point in $[0, 1]$ there will exist at least one trial point generated by the $GOSH$. □

# 5 Numerical experiments

In this section, results of some numerical experiments are presented. The new algorithm $GOSH$ has been compared with the original $DIRECT$ method [7] and its locally-biased modification $LBDirect$ proposed in [8, 9]. In order to show the usefulness of the two-phase approach, the $GOSH$ has been compared with its simplified version (called $CORE$ hereinafter) that does not apply the local phase at all and only the global phase is used.

Ten different classes of functions generated by the GKLS-generator, a free software downloadable from http://wwwinfo.deis.unical.it /∼ yaro/GKLS.html and described in [10] have been used in the experiments. This generator constructs classes of multi-dimensional and multiextremal test functions with known global and local minima: each function is obtained by a paraboloid, systematically distorted by polynomials. Each class contains 100 test functions with the same number of local minima. In order to generate a specific class, only five parameters should be defined by the user (see Table 1), and it possible to generate harder or simpler test classes very easily. For example, a more difficult test class can be obtained either by decreasing the radius $r^*$ of the attraction region of the global minimizer or by increasing the distance $d$ from the paraboloid vertex to the global minimizer. In Table 1 we can see a complete description of the 10 classes that we have used in the experiments, for a total of 1000 test functions, in dimensions $N = 2, 3, 4, 5$, and 6. For each dimension two different classes, a simple class and a hard one, have been generated. The number of local minima $m$ was taken equal to 10 and the global minimum $f^*$ was fixed to $-1$ for all the classes. In Fig. 1-left, an example of the test function no. 4 belonging to the class 1 is shown.

| Class | Difficulty | N | $f^*$ | m | $d$ | $r^*$ |
|---|---|---|---|---|---|---|
| 1 | Simple | 2 | -1.0 | 10 | 0.90 | 0.20 |
| 2 | Hard | 2 | -1.0 | 10 | 0.90 | 0.10 |
| 3 | Simple | 3 | -1.0 | 10 | 0.66 | 0.20 |
| 4 | Hard | 3 | -1.0 | 10 | 0.90 | 0.20 |
| 5 | Simple | 4 | -1.0 | 10 | 0.66 | 0.20 |
| 6 | Hard | 4 | -1.0 | 10 | 0.90 | 0.20 |
| 7 | Simple | 5 | -1.0 | 10 | 0.90 | 0.40 |
| 8 | Hard | 5 | -1.0 | 10 | 0.90 | 0.30 |
| 9 | Simple | 6 | -1.0 | 10 | 0.90 | 0.40 |
| 10 | Hard | 6 | -1.0 | 10 | 0.90 | 0.30 |

Table 1: *Description of 10 classes of randomly generated test functions used in the numerical experiments. Each class contains 100 functions.*

Let us describe the stopping rules used in the experiments. First, the tested algorithms stopped their work when the maximal number of trials $T_{max}$, equal to $10^6$ was reached. Remind, that the GKLS generates problems with known minima. This gives the possibility to use the vicinity of trials to the global minimizer as a measure of success of the work of algorithms and to construct an appropriate stoping rule. Let us denote as $y_i^*$ the global minimizer of the $i$-th function of a test class, $1 \le i \le 100$. Then, the following condition can be applied.

**Stopping criterion.** A method stops its work on the $i$-th function of a class when it generates a trial point falling in a ball $B_i$ having a radius $\rho$ and the center at the global minimizer of the $i$-th function, i.e.,

$$B_i = \{y \in R^N : \|y - y_i^*\| \le \rho\}, \quad 1 \le i \le 100. \tag{5.1}$$

In the experiments, the radius $\rho$ in (5.1) was fixed equal to $0.01\sqrt{N}$ for classes 1, 2, 3, 4, and 5, and $0.02\sqrt{N}$ for classes 6, 7, 8, 9, and 10. It should be added also that the parameter $\xi$ in (3.1) was fixed as follows

$$\xi = 10^{-4} \cdot |f_{min}(k)|,$$

where $f_{min}(k)$ is the current best function value. This choice has been considered by many authors (see [8, 9]), in particular, it has been used in the $DIRECT$ method [7] with the most robust results. For this reason, in our experiments the same value was used, as well. Notice that for the $DIRECT$ and $LBDirect$ methods it is recommended (see, e.g., [7]) to verify stopping conditions after the end of each iteration and this rule has been used in our experiments since the usage of the rule (5.1) gives an insignificant improvement only.

The value of the parameter $\delta$ in (3.5) was fixed equal to $10^{-4}$ for classes 1 and 2, $10^{-7}$ for classes 3 and 4, $10^{-9}$ for the class 5, $10^{-10}$ for classes 6 and 7, $10^{-11}$ for classes 8, 10 and equal to $10^{-12}$ for the class 9. The parameter $\delta'$ in Step 3 of the algorithm $GOSH$ was chosen equal to $\delta$.

In the algorithms $GOSH$ and $CORE$, an $M$-approximation of the Peano curve has been considered. In particular the level $M$ of the curve must be chosen taking

in mind the constraint $NM < K$, where $N$ is the dimension of the problem and $K$ is the number of digits in the mantissa depending on the computer that is used for the implementation (see [44] for more details). In our experiments we had $K = 52$, thus the value $M = 10$ has been used for classes 1–8 and $M = 8$ for classes 9 and 10.

In the $GOSH$ algorithm we must fix the parameters $IglobMax$ and $IlocMax$, in Step 1.1 and Step 1.2, that specify the maximal allowed number of iterations executed on the global and local phase, respectively, before making the general security iteration, in which the nondominated intervals in the entire domain are selected. Different choices of these parameters can affect the speed of the search towards the global solution. For this reason, a sensitivity analysis with 6 different values of the parameters $IglobMax$ and $IlocMax$ for each class has been executed. The obtained results are shown in Table 2. For each class the average and the maximal number of function evaluations calculated for all the 100 functions is reported. The best results are shown in bold.

Table 3 shows results of experiments comparing the behavior of the $GOSH$ method with the algorithms $CORE$, $DIRECT$, and $LBDirect$ on the 10 classes of test functions. Taking into account the sensitivity analysis, the following values of the two parameters of $GOSH$ have been chosen: $IlocMax = 5$ for classes 1, 5, 8, $IlocMax = 10$ for classes 4, 6, 7 and $IlocMax = 15$ for classes 2, 3, 9, 10. $IglobMax$ was fixed equal to 5 for classes 1, 2, 3, 5, 8, 9, $IglobMax = 15$ for the class 10 and equal to 20 for classes 4, 6 and 7. The values of these two parameters corresponding to the best result in relation to the column "Max" of Table 2 have been chosen.

Table 3 illustrates results of experiments with all the 10 classes and the four methods. Notice that in the column "Average" the symbol " $>$ " means that, after performing $T_{max}$ iterations, the global minimum has not been found for all functions of the class. The column "Max" reports the maximum number of function evaluations required to satisfy the stopping criterion for all the 100 functions of the class: the notation $1000000(i)$ means that after evaluating 1000000 trials, the method was not able to find the global solution for "$i$" functions of the considered class. The best results are shown in bold.

Finally, in Fig. 5 the behavior of the four methods for the function no. 55 of the class 2 is shown. In the first row Figure 5 (a) shows 1541 trials generated by DIRECT to find the global minimum of the problem and (b) 2281 trials produced by the LBDirect. In the second row Figure 5 (c) shows 597 trial points calculated by the $CORE$ and (d) 269 produced by the $GOSH$ algorithm to solve the same problem. Trial points chosen by the "local-phase" strategy are shown in red.

# 6 A brief conclusion

The problem of global minimization of a multi-dimensional, non-differentiable, and multiextremal function satisfying the Lipschitz condition over a hyperinterval, with an unknown Lipschitz constant has been considered in this paper. An approach based on the reduction of the dimension by using numerical approximations to

| N | IlocMax | IglobMax | Average | | Maximum | |
|---|---|---|---|---|---|---|
| | | | Simple class | Hard class | Simple class | Hard class |
| 2 | 5 | 5 | **180.70** | **560.00** | **521** | 1691 |
| | 5 | 15 | 191.16 | 565.32 | 1009 | 3345 |
| | 10 | 5 | 184.50 | 563.10 | 531 | 1683 |
| | 15 | 5 | 184.50 | 563.10 | 531 | **1683** |
| | 15 | 15 | 194.36 | 569.29 | 1017 | 3337 |
| | 10 | 20 | 197.96 | 568.44 | 1199 | 3367 |
| 3 | 5 | 5 | **895.12** | 1733.94 | 3895 | 7335 |
| | 5 | 15 | 930.55 | **1683.20** | 6389 | 6651 |
| | 10 | 5 | 917.52 | 1745.14 | 3879 | 7337 |
| | 15 | 5 | 920.44 | 1784.74 | **3839** | 7347 |
| | 15 | 15 | 961.82 | 1698.24 | 6379 | 6655 |
| | 10 | 20 | 977.94 | 1693.02 | 6769 | **6589** |
| 4 | 5 | 5 | 8904.92 | 18523.44 | **139409** | 207665 |
| | 5 | 15 | 10074.92 | 17625.32 | 243635 | 197053 |
| | 10 | 5 | **8892.94** | 18553.14 | 139469 | 207675 |
| | 15 | 5 | 8904.48 | 18541.28 | 139465 | 207589 |
| | 15 | 15 | 10084.68 | 17633.54 | 243417 | 197119 |
| | 10 | 20 | 10956.02 | **17466.18** | 309549 | **194499** |
| 5 | 5 | 5 | 6437.50 | 18154.77 | 37829 | **107637** |
| | 5 | 15 | 6441.84 | **18108.82** | 38319 | 121363 |
| | 10 | 5 | **6063.46** | 18166.36 | 29319 | 107749 |
| | 15 | 5 | 6434.54 | 18361.00 | 37837 | 107757 |
| | 15 | 15 | 6437.98 | 18158.98 | 38277 | 122413 |
| | 10 | 20 | 6130.40 | 18401.63 | **27113** | 157107 |
| 6 | 5 | 5 | 25271.45 | 99318.66 | 151651 | 565015 |
| | 5 | 15 | 26968.77 | 104292.00 | 299723 | 538787 |
| | 10 | 5 | 25348.27 | 99285.62 | 150357 | 565231 |
| | 15 | 5 | **25265.09** | **99262.50** | **149281** | 565031 |
| | 15 | 15 | 27007.13 | 104281.72 | 299541 | **538751** |
| | 10 | 20 | 28276.60 | 109029.94 | 373875 | 616875 |

Table 2: *Results of the sensitivity analysis. The best values are shown in bold.*

space-filling curves in order to pass from the original Lipschitz multi-dimensional problem to a univariate one satisfying the Hölder condition has been used. It has been shown that it is possible to organize a simultaneous work with multiple estimates of the Hölder constant. Such a kind of techniques has been proposed for Lipschitz optimization in 1994 in [19] and for a long time created difficulties in the framework of Hölder global optimization. A geometric technique working with a number of possible Hölder constants chosen from a set of values varying from zero to infinity has been proposed and an accelerating "two-phase" technique that performs a smart balancing of the local and global information has been introduced. Conditions ensuring convergence of the method *GOSH* to the global minimizers have been established. Extensive numerical experiments executed on 1000 test functions have shown a very promising performance of the proposed algorithm with respect to its direct competitors, in particular for hard problems. Thus, one of the mostly abstract mathematical objects – space-filling curves – have been used to develop a

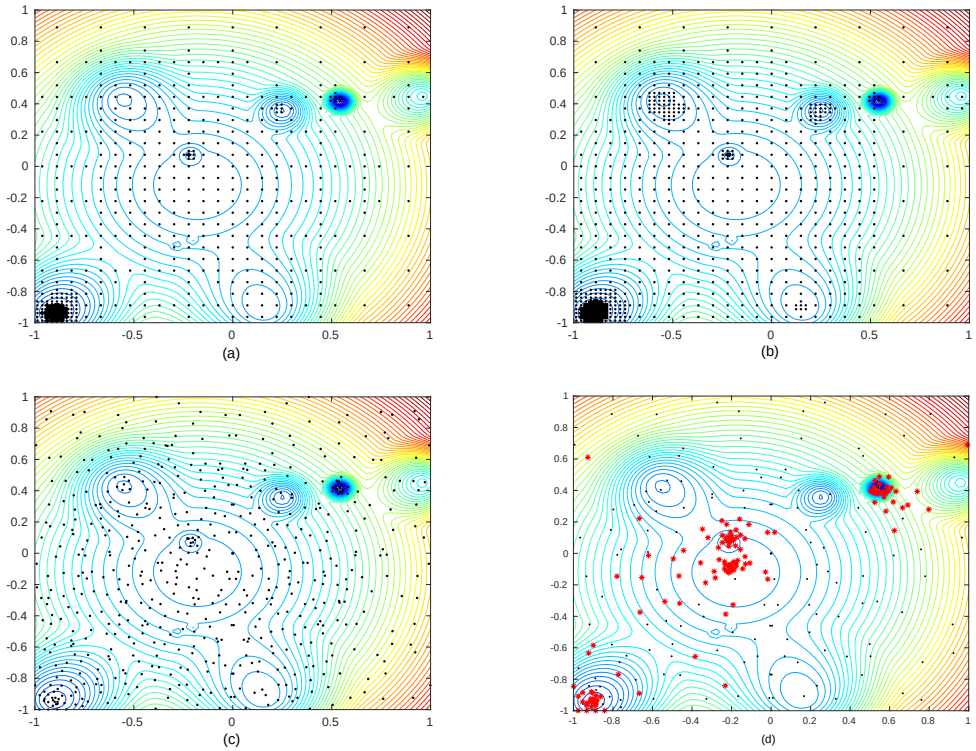


Figure 5: *Function no.55, class 2. (a): 1541 trials generated by DIRECT and (b) 2281 by LBDirect. (c): 597 trials calculated by CORE and (d) 257 produced by the GOSH. Trial points chosen by the "local-phase" strategy are shown in red by the symbol "*".*

practical derivative-free global optimization algorithm that can be successfully used in numerical computations.

**Acknowledgement**. The authors thank the unknown reviewers for their very useful comments that have allowed the authors to improve the manuscript.

The research of Ya.D. Sergeyev was supported by the Russian Science Foundation, project No 15-11-30022 "Global optimization, supercomputing computations, and applications".

# References

[1] K. A. Barkalov and V. P. Gergel. Parallel global optimization on GPU. *J. Global Optim.*, 66(1):3–20, 2016.

[2] A. R. Butz. Space filling curves and mathematical programming. *Information and Control*, 12(4):313–330, 1968.

| Class | Average number of trials | | | | Maximal number of trials | | | |
|---|---|---|---|---|---|---|---|---|
| | DIRECT | LBDirect | CORE | GOSH | DIRECT | LBDirect | CORE | GOSH |
| 1 | 208.54 | 304.28 | **174.24** | 180.70 | 1159 | 2665 | 565 | **521** |
| 2 | 1081.42 | 1291.70 | 622.60 | **563.10** | 3201 | 4245 | 1749 | **1683** |
| 3 | 1140.68 | 1893.02 | 1153.64 | **920.44** | 13369 | 20779 | 5267 | **3839** |
| 4 | >42334.36 | 5245.72 | 2077.60 | **1693.02** | 1000000(4) | 32603 | 9809 | **6589** |
| 5 | >47768.28 | 21932.94 | 10628.86 | **8904.92** | 1000000(4) | 179383 | 162183 | **139409** |
| 6 | >95908.99 | 74193.53 | 25875.16 | **17466.18** | 1000000(7) | 372633 | 319493 | **194499** |
| 7 | >33878.09 | 31955.06 | 7306.04 | **6130.40** | 1000000(3) | 146623 | 36819 | **27113** |
| 8 | >149578.61 | >93876.77 | 28391.70 | **18154.77** | 1000000(13) | 1000000(1) | 153323 | **107637** |
| 9 | >244382.63 | 184266.74 | 33366.14 | **25265.09** | 1000000(23) | 873617 | 161577 | **149281** |
| 10 | >549165.37 | >441282.91 | 132415.20 | **104281.72** | 1000000(49) | 1000000(19) | 707543 | **538751** |

Table 3: *Results of numerical experiments on 1000 randomly generated test functions. In the column "Average" the symbol " > " means that, after performing $T_{max}$ iterations, the global minimum has not been found for all functions of the class. The column "Max" reports the maximum number of function evaluations required to satisfy the stopping criterion for all the 100 functions of the class: the notation* $1000000(i)$ *means that after evaluating 1000000 trials, a method was not able to find the global solution for "*$i$*" functions of the considered class. The best results are shown in bold.*


[3] J. M. Calvin and A. Žilinskas. One-dimensional p-algorithm with convergence rate $o(n^{-3+\delta})$ for smooth functions. *J. of Optimization Theory and Applications*, 106(2):297–307, 2000.

[4] Yu. G. Evtushenko and M. Posypkin. A deterministic approach to global box-constrained optimization. *Optimization Letters*, 7(4):819–829, 2013.

[5] D. Famularo, P. Pugliese, and Ya. D. Sergeyev. A global optimization technique for checking parametric robustness. *Automatica*, 35:1605–1611, 1999.

[6] D. E. Finkel and C. T. Kelley. Additive scaling and the DIRECT algorithm. *J. Global Optim.*, 36(4):597–608, 2006.

[7] M. J. Gablonsky. DIRECT v2.04 fortran code with documentation. Technical report, http://www4.ncsu.edu/ ctk/SOFTWARE/DIRECTv204.tar.gz, 2001.

[8] M. J. Gablonsky. Modifications of the DIRECT algorithm. Technical report, Ph.D thesis, North Carolina State University, Raleigh, NC, 2001.

[9] M. J. Gablonsky and C. T. Kelley. A locally-biased form of the DIRECT algorithm. *J. Global Optim.*, 21:27–37, 2001.

[10] M. Gaviano, D. E. Kvasov, D. Lera, and Ya. D. Sergeyev. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans. Math. Software*, 29(4):469–480, 2003.

[11] V. P. Gergel, V. A., and A. V. Gergel. Adaptive nested optimization scheme for multidimensional global search. *J. Global Optim.*, 66(1):35–51, 2016.

[12] V. P. Gergel, V. A. Grishagin, and R. A. Israfilov. Local tuning in nested scheme of global optimization. *Procedia Computer Science*, 51:865–874, 2015. (International Conference on Computational Science ICCS 2015 – Computational Science at the Gates of Nature).

[13] J. W. Gillard and D. E. Kvasov. Lipschitz optimization methods for fitting a sum of damped sinusoids to a series of observations. *Stat. and Its Interface*, 10(1):59–70, 2016.

[14] E. Gourdin, B. Jaumard, and R. Ellaia. Global optimization of Hölder functions. *J. Global Optim.*, 8:323–348, 1996.

[15] V. A. Grishagin and R. A. Israfilov. Global search acceleration in the nested optimization scheme. In *AIP Conf. Proc.*, volume 1738, pages 400010:1–4. 2016.

[16] V. A. Grishagin, R. A. Israfilov, and Ya. D. Sergeyev. Convergence conditions and numerical comparison of global optimization methods based on dimensionality reduction schemes. *Applied Mathematics and Computation*, 318:270–280, 2018.

[17] R. Horst and P. M. Pardalos, editors. *Handbook of Global Optimization*, volume 1. Kluwer Academic Publishers, Dordrecht, 1995.

[18] R. Horst and H. Tuy. *Global Optimization – Deterministic Approaches*. Springer–Verlag, Berlin, 1996.

[19] D. R. Jones, C. D. Perttunen, and B. E. Stuckman. Lipschitzian optimization without the lipschitz constant. *J. of Optimization Theory and Applications*, 79:157–181, 1993.

[20] D. E. Kvasov, C. Pizzuti, and Ya. D. Sergeyev. Local tuning and partition strategies for diagonal GO methods. *Numer. Math.*, 94(1):93–106, 2003.

[21] D. E. Kvasov and Ya. D. Sergeyev. Lipschitz global optimization methods in control problems. *Automation and Remote Control*, 74(9):1435–1448, 2013.

[22] D. E. Kvasov and Ya. D. Sergeyev. Deterministic approaches for solving practical black-box global optimization problems. *Advances in Engineering Software*, 80:58–66, 2015.

[23] D. E. Kvasov and Ya. D. Sergeyev. A univariate global search working with a set of Lipschitz constants for the first derivative. *Optimization Letters*, 3(2):303–318, 2009.

[24] D. Lera and Ya. D. Sergeyev. Global minimization algorithms for Hölder functions. *BIT*, 42(1):119–133, 2002.

[25] D. Lera and Ya. D. Sergeyev. Acceleration of univariate global optimization algorithms working with Lipschitz functions and Lipschitz first derivatives. *SIAM J. Optimization*, 23(1):508–529, 2013.

[26] D. Lera and Ya. D. Sergeyev. Deterministic global optimization using space-filling curves and multiple estimates of Lipschitz and Hölder constants. *Communications in Nonlinear Science and Numerical Simulation*, 23:328–342, 2015.

[27] D. Lera and Ya. D. Sergeyev. An information global minimization algorithm using the local improvement technique. *J. Global Optim.*, 48(1):99–112, 2010.

[28] D. Lera and Ya. D. Sergeyev. Lipschitz and Hölder global optimization using space-filling curves. *Appl. Numer. Maths.*, 60:115–129, 2010.

[29] G. Liuzzi, S. Lucidi, and V. Piccialli. A direct-based approach exploiting local minimizations for the solution for large-scale global optimization problem. *Computational Optimization and Applications*, 45(2):353–375, 2010.

[30] G. Liuzzi, S. Lucidi, and V. Piccialli. A partition-based global optimization algorithm. *J. Global Optim.*, 48(1):113–128, 2010.

[31] R. Paulavičius, L. Chiter, and J. Žilinskas. Global optimization based on bisection of rectangles, function values at diagonals, and a set of Lipschitz constants. *J. Global Optim.*, 2017. In press.

[32] R. Paulavičius, Ya. D. Sergeyev, D. E. Kvasov, and J. Žilinskas. Globally-biased DISIMPL algorithm for expensive global optimization. *J. Global Optim.*, 59(2-3):545–567, 2014.

[33] R. Paulavičius and J. Žilinskas. *Simplicial Global Optimization*. SpringerBriefs in Optimization. Springer, New York, 2014.

[34] J. D. Pintér. *Global Optimization in Action (Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications)*. Kluwer Academic Publishers, Dordrecht, 1996.

[35] J. D. Pintér. Global optimization: software, test problems, and applications. In P. M. Pardalos and H. E. Romeijn, editors, *Handbook of Global Optimization*, volume 2, pages 515–569. Kluwer Academic Publishers, Dordrecht, 2002.

[36] S. A. Piyavskij. An algorithm for finding the absolute extremum of a function. *USSR Comput. Math. Math. Phys.*, 12(4):57–67, 1972. (In Russian: Zh. Vychisl. Mat. Mat. Fiz., 12(4) (1972), pp. 888–896).

[37] F. P. Preparata and M. I. Shamos. *Computational Geometry: An Introduction*. Springer–Verlag, New York, 1993.

[38] H. Sagan. *Space-Filling Curves*. Springer, New York, 1994.

[39] Ya. D. Sergeyev. An information global optimization algorithm with local tuning. *SIAM J. Optimization*, 5(4):858–870, 1995.

[40] Ya. D. Sergeyev. A one-dimensional deterministic global minimization algorithm. *Comput. Math. Math. Phys.*, 35(5):705–717, 1995.

[41] Ya. D. Sergeyev, P. Daponte, D. Grimaldi, and A. Molinaro. Two methods for solving optimization problems arising in electronic measurements and electrical engineering. *SIAM J. Optim.*, 10(1):1–21, 1999.

[42] Ya. D. Sergeyev and V. A. Grishagin. Sequential and parallel algorithms for global optimization. *Optim. Methods Softw.*, 3:111–124, 1994.

[43] Ya. D. Sergeyev and D. E. Kvasov. *Deterministic Global Optimization: An Introduction to the Diagonal Approach*. Springer, New York, 2017.

[44] Ya. D. Sergeyev, R. G. Strongin, and D. Lera. *Introduction to Global Optimization Exploiting Space-Filling Curves*. SpringerBriefs in Optimization. Springer, New York, 2013.

[45] R. G. Strongin. *Numerical Methods in Multiextremal Problems: Information–Statistical Algorithms*. Nauka, Moscow, 1978. (In Russian).

[46] R. G. Strongin and Ya. D. Sergeyev. Global optimization: fractal approach and non-redundant parallelism. *J. Global Optim.*, 27:25–50, 2003.

[47] R. G. Strongin and Ya. D. Sergeyev. *Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms*. Kluwer Academic Publishers, Dordrecht, 2000. (2nd ed., 2012; 3rd ed., 2014, Springer, New York).

[48] A. Žilinskas. On similarities between two models of global optimization: Statistical models and radial basis functions. *J. Global Optim.*, 48(1):173–182, 2010.

[49] A. Žilinskas and J. Žilinskas. Parallel hybrid algorithm for global optimization of problems occurring in MDS-based visualization. *Computers & Mathematics with Applications*, 52(1-2):211–224, 2006.

[50] A. Žilinskas and J. Žilinskas. A hybrid global optimization algorithm for nonlinear least squares regression. *J. Global Optim.*, 56(2):265–277, 2013.

[51] A. A. Zhigljavsky. *Theory of Global Random Search*. Kluwer Academic Publishers, Dordrecht, 1991.

[52] A. A. Zhigljavsky and A. Žilinskas. *Stochastic Global Optimization*. Springer, New York, 2008.