Lipschitz-inspired HALRECT Algorithm for Derivative-free Global Optimization

Linas Stripinis · Remigijus Paulavičius

Received: date / Accepted: date

Abstract This article considers a box-constrained global optimization problem for Lipschitz-continuous functions with an unknown Lipschitz constant. Motivated by the famous DIRECT (DIviding RECTangles), a new HALRECT (HALving RECTangles) algorithm is introduced. A new deterministic approach combines halving (bisection) with a new multi-point sampling scheme in contrast to trisection and midpoint sampling used in most existing DIRECT-type algorithms. A new partitioning and sampling scheme uses more comprehensive information on the objective function. Four different strategies for selecting potentially optimal hyper-rectangles are introduced to exploit the objective function's information effectively. The original algorithm HALRECT and other introduced HALRECT variations (twelve in total) are tested and compared with the other twelve recently introduced DIRECT-type algorithms on 96 box-constrained benchmark functions from DIRECTGOLib v1.1, and 96 perturbed their versions. Extensive experimental results are advantageous compared to state-of-the-art DIRECT-type global optimization. New HALRECT approaches offer high robustness across problems of different degrees of complexity, varying from simple uni-modal and low dimensional to complex – multi-modal and higher dimensionality.

Keywords DIRECT-type algorithm \cdot Global optimization \cdot Derivative-free optimization \cdot Lipschitz optimization \cdot Sampling-based algorithm

Mathematics Subject Classification (2020) $65K05\cdot74P99\cdot78M50,\,90C99\cdot65K10$

L. Stripinis, R. Paulavičius

Vilnius University, Institute of Data Science and Digital Technologies, Akademijos 4, LT-08663 Vilnius, Lithuania E-mail: linas.stripinis@mif.vu.lt

R. Paulavičius E-mail: remigijus.paulavicius@mif.vu.lt

1 Introduction

Generally, global optimization approaches can be divided into two main classes: deterministic and stochastic [17,38]. Deterministic algorithms theoretically guarantee that at least one global optimum can be found [9], while stochastic algorithms find the solution in the probability sense [21]. Various optimization problems in science and engineering (e.g., machine learning models [2], Boeing design [3], etc.) are black-box, i.e., the analytic information about the objective and constraints functions is unavailable. Therefore, the development of derivative-free optimization has been forced by the need to optimize various and often increasingly complex problems in practice.

In this paper we consider a box-constrained potentially black-box global optimization problem

$$\min_{\mathbf{x}\in D} \quad f(\mathbf{x}),\tag{1}$$

where $f : \mathbb{R}^n \to \mathbb{R}$ is a real-valued Lipschitz-continuous function, i.e., there exists a positive constant $0 < L < \infty$, such that

$$\left| f(\mathbf{x}) - f(\mathbf{y}) \right| \le L \|\mathbf{x} - \mathbf{y}\|, \quad \forall \mathbf{x}, \mathbf{y} \in D,$$
(2)

and the feasible region is an *n*-dimensional hyper-rectangle $D = [\mathbf{a}, \mathbf{b}] = {\mathbf{x} \in \mathbb{R}^n : a_j \leq x_j \leq b_j, j = 1, ..., n}$. In a black-box optimization case, the objective function f is unknown and any information can only be obtained by evaluating the function at feasible points.

The DIRECT algorithm, developed by Jones et al. [20], is a well-known and widely used sampling-based solution technique for derivative-free global optimization with limitations in the box. An algorithm is an extension of classical Lipschitz optimization (e.g., [29, 30, 35, 36, 41, 43]), where the need to know the Lipschitz constant is eliminated. The DIRECT algorithm has also been successfully extended to solve problems with various constraints. Authors in [32] proposed an approach to tackle linearly constrained problems. Other authors [5, 7, 18, 22, 33, 34, 49] introduced DIRECT-type algorithm for generally constrained or even the for problems with hidden constraints [10, 25, 44]. In addition, DIRECT-type algorithms appear more often in the parallel environment [14, 52, 50].

A decade-old comprehensive numerical benchmarking [37] showed the encouraging performance of DIRECT-type algorithms among other derivative-free global optimization methods. Our recent extensive study [50] revealed that new and potentially better DIRECT-type algorithms are available today. In [45], we also demonstrated that even better DIRECT-type algorithms could be obtained by combining various already known candidate selection and partitioning techniques, leading to even more efficient DIRECT-type algorithms. Therefore, continuous design and development of efficient DIRECT-type algorithms is important and motivated by practice needs.

Unfortunately, DIRECT-type algorithms are not without their drawbacks. Among them, two well-known ones are [11,18,19,26,28,40]: i) delayed discovery of the globally optimal solution, especially for multi-modal and greater dimensionality problems, and ii) slow fine-tuning of the solution to high accuracy. This limits DIRECT applicability mainly to lower-dimensionality global optimization problems [19]. The first drawback is possibly determined by the original sampling scheme based on one center point per hyper-rectangle. If the hyper-rectangle containing the global solution has a bad objective value at the midpoint, it is undesirable for the selection, and his further subdivision is delayed.

To address this in [26,39], the authors introduced two different diagonal sampling schemes using two points per hyper-rectangle. In this way, new algorithms, **BIRECT** [26] and **ADC** [39], intuitively reduce the chance of this situation occurring. It would require evaluating two bad points in the hyper-rectangle containing the global optimum. In [18], the author observed that to reduce the curse of dimensionality, the division of hyper-rectangles along only one longest side instead of all has a very positive impact. Moreover, various two-phase-based approaches (see, e.g., [27,39]) and hybridized **DIRECT**-type methods (see, e.g. [16,18,28,49,23,24]) were proposed to address both these shortcomings.

This paper introduces a new HALRECT (HALving RECTangles) algorithm based on a new multi-point sampling scheme efficiently combined with halving (bisection). Each hyper-rectangle is represented by considering up to 2n + 1sampling points and halved using bisection instead of just one sampled midpoint and trisection traditionally used in most DIRECT-type algorithms. Therefore, more comprehensive information about the objective function over each hyper-rectangle is captured, especially for higher-dimensionality problems, as more sampled points are considered in selecting potentially optimal hyper-rectangles.

The rest of the paper is organized as follows. Section 2 reviews relevant existing DIRECT-type modifications and summarizes the most common selection schemes and partitioning strategies used in state-of-the-art DIRECT algorithms. A description of the new HALRECT algorithm and all its new variations is given in Section 3. The extensive numerical investigation of twelve HALRECT variations and comparison with twelve recently introduced DIRECT-type algorithms [45] using 96 box-constrained global optimization test problems and their perturbed versions from DIRECTGOLib v1.1 [51] is provided in Section 4. Finally, we conclude the paper in Section 5.

2 Related literature review

This section reviews some of the most relevant DIRECT-type modifications. We begin with a recap of the original algorithm. Reviewing other DIRECT-type algorithms, we mainly focus on the proposed candidate selection, sampling, and partitioning schemes.

2.1 Original DIRECT algorithm

The original **DIRECT** algorithm is designed for box-constrained global optimization problems. Initially, the algorithm normalizes the feasible region $D = [\mathbf{a}, \mathbf{b}]$ to a unit hyper-rectangle $\overline{D} = [0, 1]^n$ and only refers to the original space D when evaluating the objective function f. Therefore, throughout this paper, when it says that the value of the objective function is evaluated at $f(\mathbf{c})$, where the midpoint $\mathbf{c} \in \overline{D}$, it is understood that the corresponding midpoint of the original domain $(\mathbf{x} \in D)$ is used, i.e.,

$$f(\mathbf{c}) = f(\mathbf{x}), \text{ where } x_j = (b_j - a_j)c_j + a_j, j = 1, \dots, n.$$
 (3)

In each iteration, certain hyper-rectangles are identified and selected as "potentially optimal hyper-rectangles" (POH) for further investigation. DIRECT samples and evaluates the objective function at the midpoint of each POH and subdivides them (into smaller hyper-rectangles) using the trisection strategy. The selection, sampling, and subdivision procedures continue until some predefined limits have not been reached. Fig. 1 illustrates this process, showing the initialization and the first two subsequent iterations of DIRECT for the two-variable Bukin6 test problem.



Fig. 1 Two-dimensional illustration of selection, central sampling, and trisection used in the original DIRECT algorithm [20] solving the Bukin6 test problem.

Regardless of the dimension, the first evaluation of the objective function is performed at the midpoint (\mathbf{c}^1). Then, the DIRECT algorithm identifies and selects the POHs. At initialization, the selection is trivial since only one hyper-rectangle (\bar{D}^1) is available (see the left panel in Fig. 1). After selection, DIRECT samples new midpoints at positions

$$\mathbf{c}^{1} \pm \frac{1}{3} d^{\max} \mathbf{e}_{j, j} \in M, \tag{4}$$

where d^{\max} is equal to the maximum side length, M is a set of dimensions with the maximum side length, and \mathbf{e}_j is the *j*th unit vector. The algorithm uses *n*dimensional trisection, with the property that the objective function is evaluated at each hyper-rectangle only once — at a midpoint. The midpoint of the initial hyper-rectangle becomes the midpoint of the new smaller "middle" one. Suppose the selected hyper-rectangle has more than one dimension with the maximum side length (as is the case for the initial hyper-rectangle). In that case, DIRECT starts the trisection from the dimension with the lowest w_i value

$$w_j = \min\{f(\mathbf{c}^1 + \frac{1}{3}d^{\max}\mathbf{e}_j), f(\mathbf{c}^1 - \frac{1}{3}d^{\max}\mathbf{e}_j)\}, j \in M,$$
(5)

and continues to the highest [19,20]. In this way, the lower function values are placed in larger hyper-rectangles (see the middle panel in Fig. 1). If all side lengths are equal, 2n + 1 new smaller non-overlapping hyper-rectangles of n distinct sizes are created.

Unlike initialization, in subsequent iterations, the selection of POHs is not trivial, as we have more than one candidate (see the middle and right panels in Fig. 1). Therefore, the selection procedure needs to be formalized. Let the current partition in iteration k be defined as:

$$\mathcal{P}_k = \{ D_k^i : i \in \mathbb{I}_k \},\tag{6}$$

where

$$\bar{D}_{k}^{i} = [\mathbf{a}_{k}^{i}, \mathbf{b}_{k}^{i}] = \{ \mathbf{x} \in \bar{D} : 0 \le a_{k_{j}}^{i} \le x_{j} \le b_{k_{j}}^{i} \le 1, j = 1, \dots, n, \forall i \in \mathbb{I}_{k} \},$$
(7)

and \mathbb{I}_k is the index set that identifies the current partition \mathcal{P}_k . The next partition, \mathcal{P}_{k+1} , is obtained by subdividing selected POHs from the current partition \mathcal{P}_k . At the first iteration (k = 0), there is always only one candidate, $\mathcal{P}_0 = \{\bar{D}_0^1\}$, which is automatically potentially optimal. The formal requirement of potential optimality in subsequent iterations is stated in Definition 1.

Definition 1 (Original selection) Let \mathbf{c}^i denote the midpoint, $f(\mathbf{c}^i)$ objective function value $f(\mathbf{c}^i)$ obtained at the midpoint, and δ^i_k be a measure (equivalently, sometimes called distance or size) of hyper-rectangle \bar{D}^i_k . Let $\varepsilon > 0$ be a positive constant and f^{\min} be the best currently found objective function value. A hyperrectangle $\bar{D}^h_k, h \in \mathbb{I}_k$ is said to be potentially optimal if there exists some rate-ofchange (Lipschitz) constant $\tilde{L} > 0$ such that

$$f(\mathbf{c}^{h}) - \tilde{L}\delta_{k}^{h} \le f(\mathbf{c}^{i}) - \tilde{L}\delta_{k}^{i}, \quad \forall i \in \mathbb{I}_{k},$$
(8)

$$f(\mathbf{c}^{h}) - \tilde{L}\delta_{k}^{h} \le f^{\min} - \varepsilon \left| f^{\min} \right|, \qquad (9)$$

and the measure of the hyper-rectangle \bar{D}_k^i is

$$\delta_k^i = \frac{1}{2} \left\| \mathbf{b}_k^i - \mathbf{a}_k^i \right\|. \tag{10}$$

The hyper-rectangle \bar{D}_k^h is potentially optimal if the lower Lipschitz bound for the objective function computed on the left-hand side of (8) is the lowest with some positive constant \tilde{L} in the current partition \mathcal{P}_k . In (9), the parameter ε is used to protect against excessive refinement of the local minima [20,27]. Therefore, the lower Lipschitz bound on POH must be lower than the current minimum value (f^{\min}) by a considerable amount $(\geq \varepsilon | f^{\min} |)$. In [20], the authors obtained good performance using ε values ranging from 10^{-3} to 10^{-7} , and by default, the $\varepsilon = 10^{-4}$ value is suggested.

A geometrical interpretation of POH selection using Definition 1 is illustrated in the right panel of Fig. 2. Here, each hyper-rectangle is represented as a dot whose horizontal coordinate is equal to the measure of the hyper-rectangle (δ_k^i) . The vertical coordinate is equal to the value of the function at the midpoint $f(\mathbf{c}^i)$. POHs satisfy both conditions of Definition 1 and correspond to the lower-right convex hull of blue marked points in Fig. 2.



Fig. 2 Visualization of selected potentially optimal rectangles in the fifth iteration of the DIRECT algorithm solving two-dimensional $Bukin\delta$ test problem.

2.2 Brief review of candidate selection schemes

Typically, the DIRECT-type algorithms include three main steps: selection (of POHs), sampling, and partitioning (subdivision). At each iteration, a specific DIRECT-type algorithm first selects the set of POHs before sampling and subdividing them. In [45], we reviewed various improvements and new ideas introduced for the selection of POH proposed in the DIRECT literature. The three most promising ones were extracted and used to construct new DIRECT-type algorithms, combining them with four different sampling and partitioning techniques. For consistency, we give a brief description and a summary (see Table 1) of the most often used selection schemes. Section 2.3 briefly reviews sampling and partitioning techniques traditionally used in DIRECT-type algorithms.

2.2.1 Improved original selection strategy

It was observed that the original candidate selection strategy could be very inefficient on symmetric and other specific problems. There may be many POHs with the same diameter δ_k^i and objective value, leading to a drastic increase in selected POHs per iteration. To overcome this, the authors of [11] proposed an improvement by selecting only one of these many "equivalent candidates". In [19], the authors showed that such modification could significantly increase the performance of the DIRECT algorithm.

2.2.2 Aggressive Selection strategy

In [1], the authors relaxed the selection criteria of POHs and proposed an aggressive version. The main idea is to select and divide at least one hyper-rectangle from each

group of different diameters (δ_k^i) with the lowest value of the function. Definition 2 formalizes the strategy for identifying an aggressive set of potentially optimal hyper-rectangles from the current partition.

Definition 2 (Aggressive selection) Let \mathbf{c}^i , $f(\mathbf{c}^i)$, and δ^i_k be defined as in Definition 1. Let $\mathbb{I}^i_k \subseteq \mathbb{I}_k$ be the subset of indices corresponding to hyper-rectangles having the same measure (δ^i_k) . The notation \mathbb{I}^{\min}_k corresponds to the subset of hyper-rectangle indices that has the smallest measure δ^{\min}_k , while \mathbb{I}^{\max}_k has the largest measure (δ^{\max}_k) , and $\mathbb{I}_k = \mathbb{I}^{\min}_k \cup \cdots \cup \mathbb{I}^{\max}_k$.

Then for each subset, \mathbb{I}_k^i (min $\leq i \leq \max$), find hyper-rectangle(s) $\overline{D}_k^h, h \in \mathbb{I}_k^i$ with the lowest function value among all of the same measure (δ_k^i) , i.e.,

$$f(\mathbf{c}^h) \le f(\mathbf{c}^l), \quad \forall l \in \mathbb{I}_k^i.$$
 (11)

For the situation presented in Fig. 2, using Definition 2, two additional hyperrectangles from the groups where the original selection (see Definition 1) does not consider are selected. From the Lipschitz optimization point of view, such an approach may seem less favorable since it explores non-potentially optimal hyper-rectangles. There is no such positive constant \tilde{L} value, using which the lower Lipschitz bound would have the lowest values for these additional candidates selected by an aggressive strategy.

2.2.3 Improved aggressive selection strategy

In [14], the authors introduced an improvement to aggressive selection. They showed that by limiting the refinement of the search space when the measure of hyper-rectangles (δ_k^i) reached some prescribed limit δ^{limit} , the memory usage might be reduced from 10% to 70%. Therefore, the improved aggressive version can run longer without memory allocation failure. We note that in our experimental part (described in Section 4), the limit parameter (δ^{limit}) for algorithms using this selection scheme was set to the measure of a hyper-rectangle that has been subdivided 50*n* times (same as in [45]).

2.2.4 Two-step-based Pareto selection

In our recent extension, DIRECT-GL [48], we introduced a new two-step-based approach to identify the extended set of POHs, formally stated in Definition 3.

Definition 3 (Two-step Pareto selection) Find all Pareto optimal hyper-rectangles that are non-dominated on size (the higher, the better) and center point function value (the lower, the better), and all non-dominated on size and distance from the current minimum point (the closer, the better). Then take the unique union of these two identified sets of candidates.

Unlike the aggressive strategy (Definition 2), using Definition 3, hyper-rectangles from the groups where the minimum objective function value is higher than the minimum value from the larger groups are not selected. Compared to the original selection (Definition 1), using Definition 3, the set of POHs is enlarged by adding more medium-sized hyper-rectangles. In this sense, Pareto selection may be more global. Additionally, in the second step, the hyper-rectangles that are

Notation & source	Identification of POH	Final selection of POH
OS (Jones et. al, 1993)	Original Selection strategy using Definition 1.	Selects all candidates which satisfies Definition 1.
AS (Baker et. al, 2000)	$egin{array}{llllllllllllllllllllllllllllllllllll$	Selects all candidates which satisfies Definition 2.
IO (Gablonsky et. al, 2001)	Improved Original selection strategy using Definition 1.	Selects only one hyper-rectangle if there is a tie for the lowest function value in the same diameter group.
IA (He et. al, 2008)	$\begin{array}{ll} Improved & Aggressive \\ selection & strategy & using \\ Definition 2, & but & limiting \\ the selection of candidates \\ to some & prescribed & limit \\ (\delta^{limit}). \end{array}$	Selects only one hyper-rectangle if there is a tie for the lowest function value in the same diameter group and $\delta_k^i \ge \delta^{\text{limit}}$.
GL (Stripinis et. al, 2018)	Two-step-based (Global- Local) Pareto selection using Definition 3.	Selects only one hyper-rectangle if there is a tie for the lowest function value or distance from the current minimum point.

 Table 1
 Summary of selection schemes typically used in DIRECT-type algorithms (in ascending order of the year of publication)

non-dominated with respect to the size and distance from the current minimum point are selected. Therefore, the set of POHs is enlarged with various size hyper-rectangles nearest the current minimum point.

2.3 Brief review of sampling and partitioning schemes

This subsection briefly reviews some of the primary sampling and partitioning techniques proposed in the **DIRECT** literature. A summary of them is given in Table 2, including illustrative examples.

2.3.1 Hyper-rectangular partitioning based on $1\text{-}dimensional\ trisection\ and\ center\ sampling$

In [18], Jones proposed a revised version of the original DIRECT algorithm. One of the main algorithmic changes was made to the partitioning scheme. The author suggested trisecting selected POHs only along the longest side (coordinate). If there are several equal longest sides, the dimension that has been split the fewest times during the search procedure is selected. If there is a tie on the latter criterion, the lowest indexed dimension is selected. In [19], the authors experimentally justified that such modification can significantly improve the performance of the original DIRECT algorithm.

Notation & Source	Partitioning scheme	Sampling scheme	Illustrative example
N-DTC (Jones et. al, 1993)	Hyper-rectangular partitioning based on N-D imensional T risection.	Sampling points are located at the Center points of each hyper- rectangle.	
1-DTC (Jones et. al, 2001)	Hyper-rectangular partitioning based on 1-D imensional T risection.	Sampling points are located at the Center points of each hyper- rectangle.	
1-DTDV (Sergeyev et. al, 2006)	Hyper-rectangular partitioning based on 1-D imensional T risection.	Sampling points are located at two Diagonal Vertices of each hyper-rectangle.	
1-DTCS (Paulavišius et. al, 2014)	Simplicial partitioning based on 1- Dimensional Trisection.	Sampling points are located at the Center points of each Simplex.	
1-DBVS (Paulavišius et. al, 2014)	Simplicial partitioning based on 1- Dimensional Bisection.	Sampling points are located at Vertices of each Simplex.	
1-DBDP (Paulavišius et. al, 2016)	Hyper-rectangular partitioning based on 1-D imensional B isection.	SamplingpointsarelocatedatDiagonalsPointsequidistantbetweenthemselvesanddiagonal's vertices.	$\begin{array}{c c} \bullet & \circ & \circ \\ \bullet & \circ & \circ & \circ \\ \bullet & \bullet & \circ & \circ \\ \bullet & \bullet & \circ & \circ \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet &$

 Table 2
 Summary of sampling and partitioning schemes typically used in the DIRECT-type algorithms (in ascending order of the year of publication)

2.3.2 Hyper-rectangular partitioning based on 1-dimensional trisection and sampling on diagonal vertices

Adaptive diagonal curves (ADC) based algorithm was proposed in [39]. Independently of the dimension, ADC evaluates the objective function $f(\mathbf{x})$ on two vertices of the main diagonals. By sampling two points per hyper-rectangle, such a partitioning scheme reduces the chance that the algorithm samples two bad points in the same hyper-rectangle containing an optimal solution. Thus, better performance could be expected, especially on more complex problems. Moreover, such a scheme has a significant advantage over center sampling methods when most solution coordinates are located on the boundaries [45]. As in the revised version of DIRECT [18], each selected POH is trisected along just one of the longest sides.

2.3.3 Simplicial partitioning based on 1-dimensional trisection/bisection and sampling at center/vertices

In DISIMPL [31], simplicial partitions are considered instead of hyper-rectangles. At the initialization step, the unit hyper-rectangle \overline{D} is partitioned into n! simplices by the standard face-to-face simplicial division based on combinatorial vertex triangulation [31]. After this, all simplices share the diagonal of the feasible region and have equal hyper-volume. In [31], two different sampling and partitioning strategies were proposed: i) evaluating the objective function at the geometric center point of the simplex and trisecting them (1-DTCS); ii) evaluating the objective function on all unique vertices of the simplex and bisecting them (1-DBVS). While simplicial partitions are very promising for symmetric [31] and problems with linear constraints [32] for box-constrained, they are less appealing as the number of initial simplices increases speedily with the number of dimensions.

2.3.4 Hyper-rectangular partitioning based on 1-dimensional bisection and sampling at two diagonal points

One of the most recent proposals, BIRECT (BIsecting RECTangles) [26], is also motivated by the diagonal partitioning strategy [39,40,42]. However, the objective function is evaluated at two points on the diagonal that are equidistant between themselves and the vertices of the diagonal. Such a sampling strategy enables the reuse of the sampling points in descendant hyper-rectangles. Moreover, the bisection is used instead of the typical trisection for diagonal-based algorithms and most DIRECT-type algorithms.

3 Description of the HALRECT algorithm

Unlike most DIRECT-type algorithms based on central sampling combined with trisection, HALRECT (HALving RECTangles) is based on a unique multi-point sampling technique combined with a halving (bisection). We first give a high-level illustration of the sampling and partitioning techniques used in the HALRECT algorithm. We illustrate them on a binary tree (see Fig. 3). Note that the value of the function was evaluated at more than one sampling point at each POH (except the initial hyper-rectangle). The experimental part shows that much more comprehensive information about the objective function over hyper-rectangles can be exploited efficiently and positively impact the algorithm's performance. In contrast to the authors of the original DIRECT, who proposed trisection, bisection combined with central sampling can also be a very efficient combination. In the following subsections, we detail the main steps of the HALRECT algorithm.

3.1 Initialization phase

Like others, the HALRECT algorithm begins by scaling the feasible region D to an n-dimensional unit hyper-rectangle \overline{D}_0^1 . It only refers to the initial space D when



Fig. 3 Sampling and partitioning techniques used in the HALRECT algorithm illustrated as a binary tree

evaluating the objective function $f(\mathbf{x})$. The selection of POH in the initialization phase is trivial, as only one candidate is available. However, in the subsequent iterations, the selection of POHs is not trivial, and Section 3.3 is devoted to formalizing this.

3.2 Partitioning and sampling scheme

Like other DIRECT-type algorithms, HALRECT samples and evaluates the objective function at midpoints (in the initial phase at $\mathbf{c}^1 \in \overline{D}_0^1$). However, unlike most other algorithms, HALRECT uses bisection instead of trisection. As a result, midpoints, after bisection, shift in different facets of the hyper-rectangle. Moreover, all these sampling points can be involved in the POHs selection process. This way, more detailed information about each hyper-rectangle is considered. Fig. 4 illustrates the selection, sampling, and subdivision procedures in the initialization and the subsequent first two iterations of HALRECT for two-and three-dimensional test problems.

Now let us formalize the sampling and partitioning schemes used in HALRECT. In iteration k, the current partition (\mathcal{P}_k) and hyper-rectangle (\bar{D}_k^i) are defined as in Eqs. (6) and (7), where \mathbb{I}_k is the index set of the current partition. Additionally, for each hyper-rectangle, we define the representative sampling index set \mathbb{H}_k^i storing the indices (i) of all sampled points (\mathbf{c}^i) within the hyper-rectangle at which the objective function has already been evaluated. We note that initially sampled midpoints, after subdivision (bisection), change their position and later are located on facets of hyper-rectangles (see Fig. 4).



Fig. 4 Illustration of selection, sampling and partitioning schemes used in the HALRECT algorithm on two-dimensional (upper part) and three-dimensional (lower part) test problems.

Using these notations, at the initial (k = 0) and the first two iterations, the current partition (\mathcal{P}_k) and the representative sampling index sets (\mathbb{H}_k^i) are

$$\begin{aligned} \mathcal{P}_0 &= \{D_0^1\}, \mathbb{H}_0^1 = \{1\}, \\ \mathcal{P}_1 &= \{\bar{D}_1^2, \bar{D}_1^3\}, \mathbb{H}_1^2 = \{1, 2\}, \mathbb{H}_1^3 = \{1, 3\}, \\ \mathcal{P}_2 &= \{\bar{D}_2^3, \bar{D}_2^4, \bar{D}_2^5\}, \mathbb{H}_2^3 = \{1, 3\}, \mathbb{H}_2^4 = \{1, 2, 4\}, \mathbb{H}_2^5 = \{1, 2, 5\}. \end{aligned}$$

Selected POHs (Section 3.3 describes the selection process) are bisected only along one coordinate with the maximum side length. Algorithm 1 describes the procedure used in HALRECT to select the branching variable, i.e., coordinate index $(br \in \{1, ..., n\})$.

Example 1 In Fig. 5, an illustration of branching variable selection is given in the HALRECT algorithm moving from the second to the third iteration. In the second iteration (k = 2), there are two POHs $(\bar{D}_2^3 \text{ and } \bar{D}_2^4)$. For \bar{D}_2^3 there is only one longest side (coordinate j = 2 with the side length $d_2^3 = 1$), therefore Algorithm 1 returns br = 2. However, for \bar{D}_2^3 , at Step 1 of Algorithm 1, both sides are equal, and therefore $\lambda_1 = \{1, 2\}$. Since the midpoint \mathbf{c}^4 is also a current minimum point (\mathbf{c}^{\min}) , after Step 2, the set $\lambda_2 = \{1, 2\}$. Finally, the coordinate with the smallest index value (br = 1) is selected in the third step and returned.

Algorithm 1 Branching coordinate index selection

Input: Selected POH (\bar{D}_k^i) , new sampling point $(\mathbf{c}^i \in \bar{D}_k^i)$, current minimum point (\mathbf{c}^{\min}) ; **Output:** Branching coordinate index (br);

1: Find all the longest sides (indices of corresponding coordinates)

$$\lambda_1 = \operatorname*{arg\,max}_{j=1,...,n} \left\{ d^i_j = | \, \bar{b}^i_j - \bar{a}^i_j \, | \right\}; // \text{See Eq. (7)}$$
(12)

2: Find the furthest coordinate(s) from \mathbf{c}^i to \mathbf{c}^{\min}

$$\lambda_2 = \underset{j \in \lambda_1}{\arg\max} \left\{ \mid c_j^i - c_j^{\min} \mid \right\};$$
(13)

3: Select the coordinate with the smallest index

$$br = \min_{j \in \lambda_2} j. \tag{14}$$

Return br.



Fig. 5 Illustration of sampling and partitioning schemes used in the HALRECT algorithm on a two-dimensional example moving from the second to the third iteration.

When the branching coordinate (br) is identified, each POH (\bar{D}_k^i) is bisected into two equal smaller hyper-rectangles \bar{D}_k^{left} and \bar{D}_k^{right} . The new midpoints $(\mathbf{c}^{\text{left}} \in \bar{D}_k^{\text{left}} \text{ and } \mathbf{c}^{\text{right}} \in \bar{D}_k^{\text{right}})$ are located at the following positions:

$$\mathbf{c}^{\text{left}} = (c_1^i, ..., c_{br}^i - \frac{d_{br}^i}{4}, ..., c_n^i), \tag{15}$$

$$\mathbf{c}^{\text{right}} = (c_1^i, ..., c_{br}^i + \frac{d_{br}^i}{4}, ..., c_n^i),$$
(16)

where $\mathbf{c}^i \in \bar{D}_k^i$. We note that naming new hyper-rectangles and midpoints as the "left" and the "right" is only relative.

Continuing in the same vein, after bisection of \bar{D}_2^3 , new midpoints are located at:

$$\mathbf{c}^{\text{left}} = \mathbf{c}^{6} = \left(c_{1}^{3}, c_{2}^{3} - \frac{d_{2}^{3}}{4}\right) = \left(\frac{3}{4}, \frac{1}{4}\right),$$
$$\mathbf{c}^{\text{right}} = \mathbf{c}^{7} = \left(c_{1}^{3}, c_{2}^{3} + \frac{d_{2}^{3}}{4}\right) = \left(\frac{3}{4}, \frac{3}{4}\right).$$

After bisection of \bar{D}_2^4 , new sampling points are located at:

$$\mathbf{c}^{\text{left}} = \mathbf{c}^8 = \left(c_1^4 - \frac{d_1^4}{4}, c_2^4\right) = \left(\frac{1}{8}, \frac{1}{4}\right),$$
$$\mathbf{c}^{\text{right}} = \mathbf{c}^9 = \left(c_1^4 + \frac{d_1^4}{4}, c_2^4\right) = \left(\frac{3}{8}, \frac{3}{4}\right).$$

The illustration of sampled search space after ten iterations using the HALRECT algorithm on Sum_of_Powers function is given in Fig. 6.



Fig. 6 The illustration of sampled points after 10 iterations of the HALRECT algorithm using two and three-dimensional Sum_of_Powers functions.

After subdivision, each POH (\bar{D}_k^i) is removed, and two new ones are added to the list that describes the current partition:

$$\mathcal{P}_{k+1} = (\mathcal{P}_k \setminus \bar{D}_k^i) \cup \bar{D}_k^{\text{left}} \cup \bar{D}_k^{\text{right}}.$$

Therefore, moving from iteration two to three, hyper-rectangles \bar{D}_2^3 and \bar{D}_2^4 are removed from the partition (\mathcal{P}_2) , and new ones are included:

$$\mathcal{P}_3 = \{ \bar{D}_3^5, \bar{D}_3^6, \bar{D}_3^7, \bar{D}_3^8, \bar{D}_3^9 \}.$$

New vectors of the representative index sets $\mathbb{H}_{k}^{\text{left}}$ and $\mathbb{H}_{k}^{\text{right}}$ are constructed based on the set \mathbb{H}_{k}^{i} corresponding to the subdivided hyper-rectangle (\bar{D}_{k}^{i}) . The following rules are used to create them:

$$\mathbb{H}_k^{\text{left}} = \{h \in \mathbb{H}_k^i : c_{br}^i \ge c_{br}^h\} \cup \{\text{left}\},\tag{17}$$

$$\mathbb{H}_{k}^{\text{right}} = \{ h \in \mathbb{H}_{k}^{i} : c_{br}^{i} \le c_{br}^{h} \} \cup \{ \text{right} \}.$$

$$(18)$$

Example 2 Let us consider the subdivided hyper-rectangle \overline{D}_2^4 , whose representative sampling index set $\mathbb{H}_2^4 = \{1, 2, 4\}$ (see Fig. 5). Then $\mathbb{H}_3^{\text{left}}$ and $\mathbb{H}_3^{\text{right}}$ consist of:

$$\mathbb{H}_{3}^{\text{left}} = \mathbb{H}_{3}^{8} = \left\{ h \in \mathbb{H}_{2}^{4} : c_{1}^{4} \ge c_{1}^{h} \right\} \cup \{8\} = \{2, 4, 8\},$$
$$\mathbb{H}_{3}^{\text{right}} = \mathbb{H}_{3}^{9} = \left\{ h \in \mathbb{H}_{2}^{4} : c_{1}^{4} \le c_{1}^{h} \right\} \cup \{9\} = \{1, 2, 4, 9\}.$$

In the following subsection, we will show how these representative index sets (\mathbb{H}_k^i) are used to select potentially optimal hyper-rectangles by taking into account up to $2 \times n + 1$ objective function values over each hyper-rectangle. But first we prove that the cardinality of \mathbb{H}_k^i cannot exceed 2n + 1.

Corollary 1 The cardinality of any representative sampling index set \mathbb{H}_k^i is less than or equal to $2 \times n + 1$, *i.e.*,

$$\max_{i \in \mathbb{I}_k, \forall k} \operatorname{card}(\mathbb{H}_k^i) \le 2 \times n + 1.$$
(19)

Proof In HALRECT, selected POHs are bisected only along one coordinate with the maximum side length. Without loss of generality, assume that br = 1, i.e., the branching (bisection) on the x_1 variable takes place. As a result, the midpoint, after bisection, shifts on the "left" and on the "right" facet of two newly created hyper-rectangles (see the middle part for two and three-dimensional illustrations in Fig. 4). This way, each subdivided hyper-rectangle cuts off the old facet and replaces it with a new one. Therefore, only one point can appear on one facet concerning the branching variable.

Throughout the search process (as the number of iterations k increases), all this will be applied to other branching variables (x_2, \ldots, x_n) too. From this follows, that the set \mathbb{H}_k^i is constructed only by points located in the hyper-rectangular facets and one midpoint. As each hyper-rectangle contains $2 \times n$ facets, the maximal number of $2 \times n + 1$ points can be included in \mathbb{H}_k^i .

3.3 Selection of potentially optimal hyper-rectangles

Since the objective function in the HALRECT algorithm is evaluated at multiple points, more comprehensive information about the objective function values can be efficiently integrated into the selection scheme. In Definition 4, we introduce four different selection schemes implemented in the new HALRECT algorithm, where the main difference is how the value \mathcal{F}_k^i is calculated (see Eqs. (23a) to (23d)).

Definition 4 (HALRECT selection) Let $\mathbf{c}^i \in \bar{D}_k^i$ denote the midpoint, $\mathbf{c}^j \in \bar{D}_k^i, j \in \mathbb{H}_k^i$ denote all sampling points (including \mathbf{c}^i) of hyper-rectangle (\bar{D}_k^i) , card (\mathbb{H}_k^i) – the cardinality of $(\mathbb{H}_k^i), \delta_k^i$ be a measure of \bar{D}_k^i , and \mathcal{F}_k^i – aggregated value based on objective function values attained at sampling point(s) whose indices belong to \mathbb{H}_k^i . Let $\varepsilon > 0$ be a positive constant and f^{\min} be the best currently found objective function value. A hyper-rectangle $\bar{D}_k^h, h \in \mathbb{I}_k$ is said to be potentially optimal if there exists some rate-of-change (Lipschitz) constant $\tilde{L} > 0$ such that

$$\mathcal{F}_{k}^{h} - \tilde{L}\delta_{k}^{h} \le \mathcal{F}_{k}^{i} - \tilde{L}\delta_{k}^{i}, \quad \forall i \in \mathbb{I}_{k},$$

$$(20)$$

$$\mathcal{F}_k^h - \tilde{L}\delta_k^h \le f^{\min} - \varepsilon \left| f^{\min} \right|, \qquad (21)$$

where the measure of the hyper-rectangle \bar{D}_k^i is

$$\delta_k^i = \left\| \mathbf{b}_k^i - \mathbf{a}_k^i \right\|,\tag{22}$$

and \mathcal{F}_k^i is defined in one of the following four ways

$$\mathcal{F}_k^i = f(\mathbf{c}^i) \tag{23a}$$

$$\mathcal{F}_k^i = \min_{j \in \mathbb{H}_k^i} f(\mathbf{c}^j) \tag{23b}$$

$$\mathcal{F}_{k}^{i} = \frac{1}{\operatorname{card}(\mathbb{H}_{k}^{i})} \sum_{j=1}^{\operatorname{card}(\mathbb{H}_{k}^{i})} f(\mathbf{c}^{j})$$
(23c)

$$\mathcal{F}_{k}^{i} = \frac{1}{2} \left(\min_{j \in \mathbb{H}_{k}^{i}} f(\mathbf{c}^{j}) + f(\mathbf{c}^{i}) \right)$$
(23d)

3.3.1 Midpoint value based selection

Definition 1 is typically used in most existing DIRECT-type algorithmic modifications to select POHs. Geometrical visualization of the selection scheme used in DIRECT was shown in Fig. 2. The same selection scheme could be directly applied using the new sampling and partitioning strategy proposed in HALRECT, as the midpoint is always included in the sampling set. It is obtained by using Eq. (23a) in Definition 4.

For the illustrative comparison of all selection schemes, we will use partitioned space in the seventh iteration of the HALRECT algorithm solving the two-dimensional *Bukin6* test function. The selected POHs using this selection scheme are shown in part (a) on the right panel of Fig. 7. Y-axis shows the objective function values attained at the midpoints $f(\mathbf{c}^i)$ of hyper-rectangles belonging to the current partition. These values can also be seen on the left panel of Fig. 7. The apparent drawback is that the midpoints of previously partitioned hyper-rectangles (see black dots on the left panel of Fig. 7) are not involved in POH selection anymore.



Fig. 7 Two-dimensional illustration (in the seventh iteration of HALRECT on Bukin6 test problem) of four different POH selection scheme variations (see Definition 4) implemented in the HALRECT algorithm and controlled by Eqs. (23a) to (23d).

3.3.2 Minimum value based selection

The second selection scheme in HALRECT is motivated by the BIRECT algorithm [26]. Instead of objective function evaluation at midpoints, the sampling and evaluation on the diagonal points equidistant between themselves and the endpoints of a diagonal are used. Then, in the selection of POHs, the minimum of these two points is used. In the HALRECT case, the best (minimum) function value is used at all the points sampled in the hyper-rectangle (\bar{D}_k^i) . It is obtained by using Eq. (23b) in Definition 4.

As more sampling points are used in the lower Lipschitz bound calculation, more information about the objective function is exploited for POH identification, likely to result in faster convergence. Therefore, on the vertical *y*-axis, instead of function values obtained at the current midpoints, the minimum values attained at all sampled points over a hyper-rectangle $(\min_{j \in \mathbb{H}_k^i} f(\mathbf{c}^j))$ are used (see part (b) on the right side of Fig. 7).

Corollary 2 For each hyper-rectangle \bar{D}_k^i the following condition holds

$$\min_{j \in \mathbb{H}_k^i} f(\mathbf{c}^j) \le f(\mathbf{c}^i) \tag{24}$$

Proof It follows directly from the definition of \mathbb{H}_k^i (see Definition 4).

3.3.3 Mean value based selection scheme

The third selection scheme implemented in HALRECT is motivated by the mean value obtained at diagonal sampling points and proposed in [39]. In the HALRECT case, the mean function value is calculated from all sampled points on the hyper-rectangle (\bar{D}_k^i) . It is obtained by using Eq. (23c) in Definition 4. Using this selection scheme, on the vertical y-axis, the mean values calculated from objective function values

attained at all sampled points over a hyper-rectangle are used (see part (c) on the right side of Fig. 7).

3.3.4 Midpoint and minimum values based selection scheme

The final selection scheme (see Eq. (23d)) implemented in HALRECT combines ideas used in Eq. (23a) and Eq. (23c) and takes the mean of these two values. On the vertical y-axis, the mean values calculated for each hyper-rectangle using two values: i) the midpoint value $f(\mathbf{c}^i)$, and ii) the minimum value $\min_{j \in \mathbb{H}_k^i} f(\mathbf{c}^j)$ are used (see part (d) on the right side of Fig. 7).

The impact of these four selection schemes on the performance of HALRECT is explored in Section 4.1.

3.3.5 Reducing the set of selected POHs

It was stated in Section 2.2 that sometimes, e.g., using Definition 1 on symmetric problems, there might exist many POHs with the same measure δ_k and objective function value, leading to a significant increase of selected "equivalent" POHs per iteration. This situation can also arise in HALRECT, mainly when Eq. (23b) is used. Then a good objective function value attained at the vertex can be shared up to 2^n hyper-rectangles.

Many authors (see, e.g., [18,19,11,45,50]) observed that selecting only one from many "equivalent" candidates can significantly increase the performance of DIRECT-type algorithms. Some authors (see, e.g., [1,11,18]) did not specify how the only candidate should be selected, while in [48,50], the authors selected a hyper-rectangle with the largest index value among them. In the HALRECT algorithm, as more sampling points per hyper-rectangle are available, we use a unique strategy to select "the most promising candidate". Specifically, we sort in ascending order the objective function values attained at the points belonging to the hyper-rectangle. Then, if there are two hyper-rectangles of the same size with the same minimum value, we compare the second smallest values and choose the hyper-rectangle with the smaller value. If the second smallest values are equal, we compare the subsequent ones.

3.4 Algorithmic steps

The complete description of the HALRECT algorithm is shown in Algorithm 2. The inputs for the algorithm are the problem (f), optimization domain (D), and one (or few) stopping criteria: required tolerance $(\varepsilon_{\rm pe})$, the maximal number of function evaluations $(M_{\rm max})$, and the maximal number of iterations $(K_{\rm max})$. After termination, HALRECT returns the value of the objective function found $(f^{\rm min})$ and the solution point $(\mathbf{x}^{\rm min})$ together with algorithmic performance measures: final tolerance – percent error (pe), the number of function evaluations (m), and the number of iterations (k).

Like almost all DIRECT-type algorithms, HALRECT performs initialization: normalization of the feasible region, initial evaluation of the objective function at the midpoint, setting initial values for performance measures, and specifying

Algorithm 2 The HALRECT algorithm

1: HALRECT(f, D, opt);

Input: Problem f, search domain D, and adjustable algorithmic parameters *opt*: tolerance $(\varepsilon_{\rm pe})$, the maximal number of function evaluations $(M_{\rm max})$ and the maximal number of iterations $(K_{\max});$ **Output:** The best objective function value f^{\min} , minimum point \mathbf{c}^{\min} , and algorithmic performance measures pe, k, m;2: Normalize the search domain D to be the unit hyper-rectangle \overline{D} ; // pe defined in Eq. (30) 3: Initialize: $\mathbf{c}^1 = (\frac{1}{2}, \dots, \frac{1}{2}), k = 1, m = 1 \text{ and } pe;$ 4: Evaluate $f^1 = f(\mathbf{c}^1)$, and set $f^{\min} = f^1$, $\mathbf{c}^{\min} = \mathbf{c}^1$, $\mathbb{H}^1_1 = \{1\}$; 5:while $pe > \varepsilon_{\rm pe}$ and $m < {\rm M}_{\rm max}$ and $k < {\rm K}_{\rm max}~{\rm do}$ Identify the set $S_k \subseteq \mathcal{P}_k$ of POHs applying Definition 4; 6: 7: for each $\bar{D}_k^j \in S_k$ do Find the branching coordinate index (br) using Algorithm 1; 8: Bisect \bar{D}_k^j into two new hyper-rectangles \bar{D}_k^{m+1} and \bar{D}_k^{m+2} ; 9: Create new midpoints \mathbf{c}^{m+1} and \mathbf{c}^{m+2} ; // see Eqs. (15) and (16) 10:Construct $\mathbb{H}^{m+1}, \mathbb{H}^{m+2};$ // see Eqs. (17) and (18) 11: Update the partition set: $\mathcal{P}_k = \mathcal{P}_k \setminus \bar{D}_k^j \cup \bar{D}_k^{m+1} \cup \bar{D}_k^{m+2'}$ 12:if $f(\mathbf{c}^{m+1}) \leq f^{\min}$ or $f(\mathbf{c}^{m+2}) \leq f^{\min}$ then 13: Update f^{\min} , \mathbf{c}^{\min} ; 14: end if 15:16: Update performance measures: k, m and pe; 17:end for 18: end while 19: Return f^{\min} , \mathbf{c}^{\min} , and algorithmic performance measures: k, m and pe.

stopping conditions (see Algorithm 2, lines 2–4). The main *while* loop (see Algorithm 2, lines 5–18) is executed until any specified stopping condition is satisfied. At the beginning of each iteration, the HALRECT algorithm identifies the set of POHs (see Algorithm 2, line 6). As noted in the previous section, the HALRECT algorithm uses four different approaches controlled by Eqs. (23a) to (23d). Then, the HALRECT algorithm bisects all POHs, samples at new midpoints of created hyper-rectangles and updates performance measures. In the end, the solution is found, and the performance measures are returned.

3.5 Convergence properties of the HALRECT algorithm

The convergence properties of DIRECT-type algorithms are broadly reviewed and investigated (see, e.g., [8,20,26,27,39]). Typically, they belong to the class of "divide the best" methods and have the "everywhere-dense" type of convergence, that is, convergence to each point of the feasible region. The continuity of the objective function (at least in the neighborhood of global minima) is the only assumption required to ensure convergence.

The convergence of HALRECT follows from a logic similar to that of other DIRECT-type algorithms. Let us state it formally in Theorem 1, when the maximal allowed number of generated trial points, or the maximal number of function evaluations, $M_{\max} \rightarrow \infty$.

Theorem 1 For any global minima $\mathbf{x}^* \in \overline{D}$ and any $\epsilon > 0$ there exists an iteration number $k_{\epsilon} \geq 1$ and a sampling point $\mathbf{c}^j \in \overline{D}_k^{i^*} \subseteq \overline{D}$, such that

$$\max_{j \in \mathbb{H}_{k}^{\times}} \{ \| \mathbf{c}^{j} - \mathbf{x}^{*} \| \} \le \epsilon.$$

$$(25)$$

Proof In the selection scheme developed in HALRECT (see Definition 4), every iteration (k) always selects at least one hyper-rectangle $\bar{D}_k^{\max} \in S_k \subseteq \mathcal{P}_k$ from the group of hyper-rectangles with the most extensive measure δ_k^{\max} (see the right panel of Fig. 7)

$$\delta_k^{\max} = \max_{i \in \mathbb{I}_k} \{\delta_k^i\}.$$
 (26)

From Eq. (26) follows, the hyper-rectangle \bar{D}_k^{\max} with the largest measure δ_k^{\max} will be bisected through the longest coordinate (see Section 3.2) in each HALRECT iteration. Since each group δ_k of distinct measures contains only a finite number of hyper-rectangles, all hyper-rectangles of the group δ_k^{\max} will be partitioned after a sufficient number of iterations.

The procedure will be repeated with a new group of the largest hyper-rectangles. As a result, after the finite number of iterations, the current partition $\mathcal{P}_k, k \geq k_{\epsilon}$ will have only hyper-rectangles measured $\delta_k^{\max} \leq \epsilon$, i.e.,

$$\|\mathbf{b}_{k}^{i^{\max}} - \mathbf{a}_{k}^{i^{\max}}\| \le \epsilon.$$
(27)

From Eq. (27) follows, the measure $\delta_k^{i^*}$ of the hyper-rectangle containing the global minimum $\mathbf{x}^* \in \bar{D}_k^{i^*}$ also does not exceed ϵ

$$\|\mathbf{b}_k^{i^*} - \mathbf{a}_k^{i^*}\| \le \epsilon.$$
(28)

Moreover, it is clear, that

$$\max_{j \in \mathbb{H}_k^{i^*}} \{ \| \mathbf{c}^j - \mathbf{x}^* \| \} \le \delta_k^{i^*}.$$

$$\tag{29}$$

Thus, from Eqs. (28) and (29) follows Eq. (25).

4 Experimental results

This section describes the numerical experiments conducted to evaluate the performance of the newly introduced HALRECT algorithm and all its modifications by comparing them with other well-known and relevant DIRECT-type approaches. In total, we examine twelve variations of HALRECT. We compared them with twelve recently introduced DIRECT-type algorithms [45] available in the most recent version of DIRECTGO v1.1.0 [46] using 96 box-constrained global optimization test problems and their perturbed versions from DIRECTGOLib v1.1 [47,51] (listed in Table 5 in Appendix A).

In our recent study [45], we stress that the optimization domains (D) for certain test problems were redesigned to eliminate the dominance of particular partitioning schemes. The exact modified domains are also considered in this paper. Note that different subsets (e.g., low dimensional problems $(n \leq 4)$, non-convex problems, etc.) of the entire set were used to deepen the

investigation. All the problems and algorithms used in this section are implemented in the Matlab R2022a environment. All computations were performed using an Intel R CoreTM i5-10400 @ 2.90GHz processor and 16 GB RAM. All algorithms were tested using a limit of $M_{max} = 10^6$ function evaluations in each run. For the 96 analytical test cases with a priori known global optima f^* , one of the used stopping criteria is based on the percent error:

$$pe = 100 \times \begin{cases} \frac{f(\mathbf{x}) - f^*}{|f^*|}, & f^* \neq 0, \\ f(\mathbf{x}), & f^* = 0, \end{cases}$$
(30)

where f^* is the known global optimum. Thus if not specified differently, tested algorithms were stopped when the percent error became smaller than the prescribed value equal to $\varepsilon_{\rm pe} = 10^{-2}$ or when the number of function evaluations exceeded the prescribed limit of 10^6 .

Testing results shown in this article are also available in digital form in the Results/JOGO2 directory of the GitHub repository [46]. The Scripts/JOGO2 directory of the same GitHub repository [46] provides the MATLAB script for cycling through all DIRECTGOLib v1.1 test problems used in this article. The script can reproduce the results presented here. In addition, they can be used to compare and evaluate newly developed algorithms.

4.1 Comparison of different selection strategies in HALRECT

In this section, the impact and comparison of three different selection schemes: Lipschitz-based (using Definition 4), improved aggressive (using Definition 2), and two-step based Pareto (using Definition 3), and four different strategies to obtain an aggregated objective function information over hyper-rectangles (controlled by Eqs. (23a) to (23d)) in the performance of HALRECT is investigated. In total, twelve different variations of HALRECT are compared.

The results obtained on the entire set of 96 DIRECTGOLib v1.1 test problems are summarized in Table 3. The best results are highlighted in bold. In the upper part of this table, the performance of HALRECT is given using four strategies to obtain the aggregated information about the objective function (\mathcal{F}_k^i) . As can be seen, there is no single superior strategy. The best average results are obtained with the first strategy based on a single midpoint value (see Eq. (23a)). However, the overall lowest number of unsolved problems (7/96) was obtained with the second strategy. It is based on the minimum value attained at all sampled points belonging to a certain hyper-rectangle (see Eq. (23b)). Moreover, it performed significantly better on average than the other strategies on low-dimensional ($n \leq 4$) problems. It can also be seen that the third strategy, based on the mean value (see Eq. (23c)), was the worst for practically all summarized cases. The best median results were obtained with the fourth strategy (see Eq. (23d)), which combines all three strategies, using the arithmetic mean of the estimates used in the first two strategies.

Our recent work [45] showed that combining existing partition and selection schemes into DIRECT-type algorithms could lead to more efficient ones. Motivated by this, we have created two different HALRECT algorithmic versions, HALRECT-IA and HALRECT-GL, where the original partition strategy is used, but the selection

scheme is changed. Specifically, in HALRECT-IA, the original Lipschitz lower bounds-based selection scheme (Definition 4) is replaced with the *improved aggressive selection* (Section 2.2.3) using newly introduced Eqs. (23a) to (23d) for the information about the objective function. Similarly, in HALRECT-GL, the original HALRECT selection scheme is replaced with a *two-step-based* (*Global-Local*) *Pareto selection* (Section 2.2.3). Consequently, the results obtained on the same testbed are summarized in the middle and bottom parts of Table 3. Comparing the influence of Eqs. (23a) to (23d) on the performance of three different versions of HALRECT, we observe that for both HALRECT-IA and HALRECT-GL, the best results for practically all cases are obtained when Eq. (23d) is used. However, in the case of HALRECT-IA and HALRECT-GL, we no longer observe that Eq. (23c) is always the worst, as was the case for HALRECT. Comparing HALRECT, HALRECT-IA, and HALRECT-GL, we observe that the lowest number of unsolved problems (2/96) is obtained using HALRECT-GL. It was the best for almost all criteria, except for the median value, where HALRECT with Eq. (23d) performed the best.

Table 3 Comparison of HALRECT versions based on three different selection schemes: Lipschitzbased (used in HALRECT), improved aggressive (used in HALRECT-IA), and two-step based Pareto (used in HALRECT-GL) and four different strategies to obtain an aggregated objective function information (controlled by Eqs. (23a) to (23d)). The performance measured as the number of function evaluations. The best results are marked in bold.

Alg.	Criteria	# of cases	Eq. (23a)	Eq. (23b)	Eq. (23c)	Eq. (23d)
	# of failed problems	96	8	7	15	12
	Median results	96	1,419	2,119	3,581	976
	Average results	96	127,562	143,909	216,933	142,403
5	Average $(n \leq 4)$	51	30,792	7 , 248	31,456	48,456
- FR	Average $(n > 4)$	45	237,918	298,952	427,839	249,953
HAI	Average (convex)	30	93,018	167, 616	236, 137	${f 83,835}$
	Average (non-convex)	66	143,263	133, 133	208,204	169,024
	Average (uni-modal)	15	62,774	159, 110	221,099	60,046
	Average (multi-modal)	81	142, 513	140 , 401	215,972	161,408
	# of failed problems	96	9	9	15	5
	Median results	96	1,826	1,880	2,737	1,581
IA	Average results	96	114,222	124,552	194,832	${\bf 62, 874}$
Ļ	Average $(n \leq 4)$	51	43,203	10,634	13,223	3 , 762
E E E	Average $(n > 4)$	45	195,668	253,895	400,948	129,952
ALR	Average (convex)	30	105,963	139,058	234,394	51, 166
Η	Average (non-convex)	66	117,975	117,958	176,849	68, 197
	Average (uni-modal)	15	65,698	127,693	250,054	62, 332
	Average (multi-modal)	81	125, 419	123,827	182,084	$\boldsymbol{63,000}$
	# of failed problems	96	5	7	5	2
	Median results	96	1,404	2,564	2,185	1,520
ECT-GL	Average results	96	64,275	107, 127	65,271	41,061
	Average $(n \leq 4)$	51	25,847	10,831	4,360	3,301
	Average $(n > 4)$	45	108,401	216,503	134,399	83,929
ALR	Average (convex)	30	40,343	79,374	13,521	7,055
H/	Average (non-convex)	66	75, 153	119,742	88,793	56, 519
	Average (uni-modal)	15	24,714	116,545	65,538	18,646
	Average (multi-modal)	81	73,405	104,953	65,209	46 , 234

Additionally, the operational characteristics [13,53] using all 96 test problems from DIRECTGOLib v1.1 are reported in Fig. 8. Operational characteristics

provide the proportion of test problems that can be solved within a given budget of function evaluations. Fig. 8 reveals that all HALRECT algorithms based on three different selection schemes and four different strategies for \mathcal{F}_k^i (Eqs. (23a) to (23d)) perform similarly when the budget given for the evaluations of objective functions is relatively small ($m \leq 1,000$). Within this budget, all versions of HALRECT could solve approximately half of the test problems. However, as the number of function evaluations increases (as more complex problems are considered), the dominance of Eq. (23d) based versions (especially HALRECT-GL) begins to emerge. At the same time, the worst results come from versions based on Eq. (23c).



Fig. 8 Operational characteristics of HALRECT, HALRECT-IA, HALRECT-GL algorithms based on Eqs. (23a) to (23d) (used in the selection scheme) on the whole set of DIRECTGOLib v1.1 test problems.

4.2 Comparison of three <code>HALRECT</code> variations vs. twelve recent <code>DIRECT-type</code> algorithms

Based on the results presented in the previous section, the three most promising variations of HALRECT algorithms (all based on Eq. (23d)) are considered and compared with twelve different DIRECT-type global optimization variations introduced in [45]. These twelve DIRECT-type algorithms have been created by newly combining three known selection schemes: i) Improved Original (IO), ii) Improved Aggressive (IA), and iii) two-step-based (Global-Local) Pareto (GL) (see Table 1), and four partitioning techniques: i) Hyper-rectangular partitioning based on N-Dimensional Trisection and objective function evaluations at Center points (N-DTC), ii) Hyper-rectangular partitioning based on 1-Dimensional Trisection and objective function evaluations at Center points (1-DTC), iii)

Hyper-rectangular partitioning based on 1-Dimensional Trisection and objective function evaluations at two Diagonal Vertices (1-DTDV), and iv) Hyper-rectangular partitioning based on 1-Dimensional Bisection and objective function evaluations at two Diagonal Points (1-DBDP) (see Table 2).

Table 4 shows the summarized comparative results on the whole set of 96 box-constrained test problems from DIRECTGOLib v1.1. In Table 4, each column corresponds to a DIRECT-type algorithm based on a different partitioning scheme. Since each partition scheme was run on 96 problems using 3 different selection methods (rows of Table 4), each DIRECT-type algorithm based on a certain partition scheme was involved in solving $3 \times 96 = 288$ problems. As previously, the best results are marked in bold. We note that the original HALRECT algorithm does not have the purpose of adapting the IO scheme designed to reduce the number of "equivalent" hyper-rectangles. As described in Section 3.3.5, the HALRECT algorithm internally uses an innovative approach to deal with such cases.

Regardless of the chosen POH selection scheme (IO, IA, GL), the smallest number of unsolved problems was achieved using the HALRECT partitioning scheme-based algorithms (HALRECT, HALRECT-IA, HALRECT-GL). Summing the results, HALRECT partitioning scheme-based algorithms did not solve (19/288) of the test cases, while the second and third best partitioning schemes (1-DBDP and N-DTC) based algorithms did not solve (28/288) and (29/282) cases accordingly. Naturally, a higher number of solved problems leads to better performance of the HALRECT partitioning scheme-based algorithms. Consequently, the three HALRECT partitioning scheme-based algorithms required approximately 31% and 36% evaluations of fever functions in comparison to the other two algorithms based on the best partition schemes (1-DBDP and N-DTC) based algorithms. The most notable difference in the HALRECT partitioning scheme was observed when the IA selection scheme was used. The HALRECT-IA algorithm required approximately 57% and 61% compared to the two best algorithms (1-DBDP-IA and 1-DTC-IA).

In different subsets of test problems, again, on average, the HALRECT partitioning scheme-based algorithms dominate the other schemes. The dominance of the HALRECT partitioning scheme can be seen especially on more complex, multi-modal, non-convex, and n > 4 test problems. Solving multi-modal problems with HALRECT partitioning scheme-based algorithms required approximately 33% and 37% evaluations of fever functions compared to the other two algorithms based on the best partition schemes (1-DBDP and N-DTC). Among the different selection schemes, the highest level of dominance has been observed using the GL selection scheme. HALRECT-GL required approximately 62% and 65% fever function evaluations compared to the other two best algorithms (1-DBDP-GL and N-DTC-GL). On a subset of non-convex test cases, HALRECT partitioning-based algorithms required approximately 38% and 55% fever function evaluations than the other two best algorithms (1-DBDP and N-DTC). Once again, the HALRECT-GL version has shown even more outstanding performance and outperformed the second-best algorithm 1-DBDP-GL by approximately 59% of fever function evaluations.

Apart from the convex test problems, the advantage of HALRECT partitioning scheme-based algorithms is lesser on more straightforward test problems. For the $n \leq 4$ optimization test instances, HALRECT partitioning-based algorithms required approximately 4% and 24% fever function evaluations than the other two best 1-

ions based on Eq. (23d) vs. twelve DIRECT-type	
in seconds) of three HALRECT versi	e best results are marked in bold.
ion evaluations and the execution time () on DIRECTGOLib v1.1 test problems. Th
Table 4 The number of funct.	algorithms (introduced in [45])

algorithms (introduc	ed in [45])	on DIRECTG	OLib v1.1 t	est problen	as. The best	results are m	arked in bold	_:			
Criteria / Algorithms	# of cases		F	inction evaluat	tions			Execu	tion time (in s	seconds)	
	10 H	HALRECT	N-DTC-IO	1-DTC-IO	1-DBDP-IO	1-DTDV-IO	HALRECT	N-DTC-IO	1-DTC-IO	1-DBDP-IO	1-DTDV-IO
# of failed problems	96	12	12	18	12	21	I	I	I	I	I
Average results	96	142,403	142, 277	211,463	146, 133	227, 455	652.97	184.77	435.49	306.43	9, 533.28
A verage $(n \leq 4)$	51	48,456	43, 832	42,633	41,602	41,990	317.96	180.41	321.23	169.54	1, 730.39
Average $(n \ge 4)$	45	249,953	254, 819	403, 749	265, 522	438, 574	1,039.72	193.73	572.13	465.33	18, 414.99
Average (convex)	30	83, 835	111,817	170, 675	80,490	171,868	105.17	99.51	292.37	89.43	7,204.96
Average (non-convex)	66	169,024	156, 122	230,004	175, 971	252, 722	901.97	223.53	500.55	405.06	10, 591.60
Average (uni-modal)	15	60,046	60, 100	57, 360	62,016	111, 547	55.23	27.32	100.38	62.29	4,800.28
Average (multi-modal)	81	161,408	161, 240	247,026	165, 545	254, 203	790.91	221.11	512.83	362.77	10,625.50
Median results	96	946	771	1,198	953	847	0.68	0.17	0.27	0.30	0.75
Criteria / Algorithms	# of cases	HALRECT-IA	N-DTC-IA	1-DTC-IA	1-DBDP-IA	1-DTDV-IA	HALRECT-IA	N-DTC-IA	1-DTC-IA	1-DBDP-IA	1-DTDV-IA
# of failed problems	96	с.	13	13	11	18	I	I	I	I	I
Average results	96	62, 874	172,805	160, 691	146, 887	202,694	18.67	21.43	69.06	33.59	8,580.81
A verage $(n \leq 4)$	51	3, 762	25,968	23,638	45,643	9, 785	1.09	2.74	4.10	14.03	16.80
A verage $(n > 4)$	45	129,952	339, 791	316, 541	262, 640	421, 539	38.61	42.67	142.76	56.07	18, 287.05
Average (convex)	30	51, 166	149, 711	126,030	109, 374	153, 594	19.06	17.65	51.27	24.62	7,204.24
Average (non-convex)	66	68, 197	183, 302	176,446	163, 939	225,012	18.49	23.15	77.14	37.67	9,206.52
Average (uni-modal)	15	62, 332	108,068	78, 226	73,957	111,805	20.90	9.61	30.50	15.28	4,800.43
Average (multi-modal)	81	63,000	187, 744	179, 722	163, 717	223,668	18.15	24.15	77.95	37.82	9,453.20
Median results	96	1,581	7,608	1,287	2,108	1,586	0.41	0.41	0.21	0.21	0.53
Criteria / Algorithms	# of cases	HALRECT-GL	N-DTC-GL	1-DTC-GL	1-DBDP-GL	1-DTDV-GL	HALRECT-GL	N-DTC-GL	1-DTC-GL	1-DBDP-GL	1-DTDV-GL
# of failed problems	96	6	4	υ	n	n	I	I	I	1	I
Average results	96	41,061	71,488	62, 475	65,442	71, 319	16.31	9.10	38.12	21.25	3,907.79
Average $(n \leq 4)$	51	3, 301	9,675	7,073	41,300	5,772	0.63	1.08	1.09	14.33	10.68
A verage $(n > 4)$	45	83, 929	141, 753	125, 417	93,714	145, 733	34.09	18.21	80.11	29.41	8, 324.75
Average (convex)	30	7,055	55, 320	45, 520	42, 326	8,950	1.48	6.94	22.66	13.39	20.79
Average (non-convex)	66	56, 519	78,837	70, 182	75,949	99,669	23.05	10.08	45.14	24.82	5,674.61
Average (uni-modal)	15	18,646	28,478	12,624	23,300	25,796	4.78	2.09	2.28	2.94	4, 310.88
Average (multi-modal)	81	46, 234	81, 183	73,979	75,398	81, 825	18.97	10.71	46.39	25.48	3,814.77
Median results	96	1.520	1.848	096	2.042	775	0.42	0.19	0.17	0.23	0.39

 $\label{eq:lipschitz-inspired HALRECT Algorithm for Derivative-free Global Optimization$

DTDV and 1-DTC partitioning-technique-based algorithms. However, looking at individual algorithms, the most efficient HALRECT-GL algorithm outperformed the second-best 1-DTDV-GL by requiring approximately 43% fewer objective function evaluations. Similar trends persist for uni-modal test problems.

The median value is the only criterion for which HALRECT partitioning-based algorithms were not dominant. Based on the median values, 1-DTDV and 1-DTC algorithms appear to be the most effective and can solve at least half of the problems with the best performance.

Based on the number of function evaluation criteria among the selection schemes, the best overall performance was achieved using two-step-based Pareto selection (GL). All partitioning strategies combined with the latter POH selection scheme solved the largest number of test problems and showed the best performance, especially on more complex ones. The best combination, out of 15 tested, proved to be the HALRECT-GL algorithm, the second-best 1-DTC-GL, and the third-best HALRECT-IA.

Based on execution times, the fastest performing partitioning scheme is N-DTC. On average, the N-DTC required approximately 41% of fever execution times than the second fastest partition strategies (1-DBDP). It is not surprising since the N-DTC partitioning scheme subdivides the hyper-rectangle through all the largest side lengths, resulting in more function evaluations but fewer expensive computations, like POH selection. Overall, the proposed HALRECT partitioning scheme ranks only fourth in speed. Additional calculations hampered the performance of the algorithm. However, due to their exceptional performance and a small number of failures, the HALRECT-GL and HALRECT-IA algorithms rank second and third in terms of running speed, behind only the N-DTC-GL algorithm. Finally, the situation in favor of the HALRECT algorithm will be even more promising when the values of the objective functions are more expensive. In the case studied, the test functions are cheap.

Finally, operational characteristics in Fig. 9 show the behavior of all fifteen algorithms on all box-constrained test problems from DIRECTGOLib v1.1. When a given budget of function evaluations is low ($M_{\rm max} \leq 1,000$), all algorithms perform similarly regardless of the partitioning scheme. All algorithms solved approximately 60% of the test problems within this relatively small budget. However, when the maximum budget for function evaluations increased ($M_{\rm max} > 1,000$), the algorithms based on the HALRECT partitioning strategy combined with IA and GL selection schemes showed the best performance.

4.3 Investigating the impact of the domain perturbation

In investigating different partitioning techniques, one method may be lucky, because the partitioning approach in the initial steps naturally samples near the solution. In such situations, the location of the solution may favor one partitioning strategy over another. This section investigates the robustness of the partitioning approaches, especially the newly introduced HALRECT, to slight perturbations of the domain. This work extends our similar experiments described in [45], where we identified test problems for which a particular partitioning scheme (regardless of the selection strategy) had a clear dominance, possibly due to the conveniently defined variable bounds. Using the following



Fig. 9 Operational characteristics of three new HALRECT variations (based on HALRECT partitioning scheme) vs. twelve DIRECT-type algorithms (introduced in [45]) on the whole set box-constrained test problems from DIRECTGOLib v1.1.

rule, we perturbed the initial domain $(D = [\mathbf{a}, \mathbf{b}])$ for all box-constrained test problems from DIRECTGOLib v1.1:

$$D_j^{\text{pert}} = [\min(a_j + \rho d_j, x_j^{\min}), b_j + \rho d_j]_j, j = 1, \dots n,$$
(31)

where $d_j = |b_j - a_j|$, and ρ is a percentage of the shift. The perturbed domain D^{pert} is obtained by shifting the original D (given in Table 5) by a ρ percentage. Since there is a risk that the solution may change when the domain is shifted, the calculation on the left-hand side of the bound checks that the shifted $(a_j + \rho d_j)$ coordinate is not greater than x_j^{\min} . We used two different values ($\rho = 2.5\%$ and $\rho = 5\%$) for the domain perturbation in the experimental study.

The experimental results obtained of five DIRECT-type algorithms based on different partitioning schemes combined with GL selection are illustrated in Fig. 10. The efficiency of the HALRECT-GL algorithm, when a given budget of function evaluations is low ($M_{\rm max} \leq 200$), has increased. However, when ($30,000 < M_{\rm max} < 200,000$), the performance of the HALRECT-GL algorithm slightly worsened compared to the initial results. However, the algorithm's efficiency remains the same with a large objective function evaluation budget ($M_{\rm max} > 200,000$). Algorithms based on other partitioning schemes behave similarly for different values of ρ . The percentage of solved test problems remains similar for almost all algorithms. A more noticeable difference is using a 1-DTDV-GL algorithm. When $\rho = 2.5\%$, the percentage of solved problems is reduced by ~ 5%, and when $\rho = 5\%$, it is reduced by ~ 4%.

5 Conclusion

This paper introduces a new DIRECT-type algorithm (HALRECT) for box-constrained global optimization problems. A new deterministic approach combines halving



Fig. 10 Operational characteristics of five DIRECT-type algorithms based on different partitioning schemes combined with GL selection on the whole set of box-constrained perturbed problems from DIRECTGOLib v1.1.

(bisection) with a new multi-point sampling scheme in contrast to trisection and midpoint sampling used in most existing DIRECT-type algorithms. Three selection schemes and four strategies are introduced to calculate the aggregated information of the objective function used in the selection of the candidate. In this way, twelve variations of the HALRECT algorithm are introduced and experimentally compared. Three of the most promising versions were selected and compared versus twelve recent DIRECT-type algorithms. The extensive experimental results revealed that the new algorithms based on HALRECT partitioning schemes give results comparable and often superior to these 12 DIRECT-type algorithms. Further investigation has shown that small perturbations in the domain *D* of the test problems can help the HALRECT algorithm to represent better and select POHs, which can significantly improve performance efficiency.

Code availability

All implemented versions of the HALRECT algorithm are available at the GitHub repository: https://github.com/blockchain-group/DIRECTGO and can be used under the MIT license. We welcome contributions and corrections to this work.

Data statement

- GitHub: https://github.com/blockchain-group/DIRECTGOLib/tree/v1.1,
- Zenodo: https://doi.org/10.5281/zenodo.6491951,

and used under the MIT license. We welcome contributions and corrections to this work.

A DIRECTGOLib v1.1 library

A summary of all used box-constrained optimization problems from DIRECTGOLib v1.1[51,47] and their properties are given in Table 5. [45] Test problems with the α symbol indicate that the non-default domain D was used for the test problem. The modified domain D was taken from the [45] study for all the α symbol-marked test problems. Here, the main features are reported: problem number (#), name of the problem, source, dimensionality (n), optimization domain (D), problem type, and the known minimum (f^*). Some of these test problems have several variants, e.g., *Bohachevsky*, *Shekel*, and some of them, like *Alpine*, *Csendes*, *Griewank*, etc., can be tested for varying dimensionality.

References

- Baker, C.A., Watson, L.T., Grossman, B., Mason, W.H., Haftka, R.T.: Parallel Global Aircraft Configuration Design Space Exploration, p. 79–96. Nova Science Publishers, Inc., USA (2001)
- Bishop, C.M., Nasrabadi, N.M.: Pattern recognition and machine learning, vol. 4. Springer, New York, NY, USA (2006)
- Booker, A.J., Dennis, J., Frank, P.D., Serafini, D.B., Torczon, V.: Optimization using surrogate objectives on a helicopter test example. In: Computational Methods for Optimal Design and Control, pp. 49–58. Springer, New York, NY, USA (1998). DOI 10.1007/ 978-1-4612-1780-0_3
- Clerc, M.: The swarm and the queen: towards a deterministic and adaptive particle swarm optimization. In: Proceedings of the 1999 Congress on Evolutionary Computation-CEC99 (Cat. No. 99TH8406), vol. 3, pp. 1951–1957 Vol. 3. IEEE, Washington, DC, USA (1999). DOI 10.1109/CEC.1999.785513
- Costa, M.F.P., Rocha, A.M.A.C., Fernandes, E.M.G.P.: Filter-based direct method for constrained global optimization. Journal of Global Optimization 71(3), 517–536 (2018). DOI 10.1007/s10898-017-0596-8
- Dixon, L., Szegö, C.: The global optimisation problem: An introduction. In: L. Dixon, G. Szegö (eds.) Towards Global Optimization, vol. 2, pp. 1–15. North-Holland Publishing Company, Amsterdam, Netherlands (1978)
- Finkel, D.E.: MATLAB source code for DIRECT. http://www4.ncsu.edu/~ctk/Finkel_ Direct/ (2004). Online; accessed: 2017-03-22
- Finkel, D.E., Kelley, C.T.: Additive scaling and the DIRECT algorithm. Journal of Global Optimization 36(4), 597–608 (2006). DOI 10.1007/s10898-006-9029-9
- Floudas, C.A.: Deterministic global optimization: theory, methods and applications, Nonconvex Optimization and Its Applications, vol. 37. Springer US, Boston, MA (1999). DOI 10.1007/978-1-4757-4949-6
- 10. Gablonsky, J.M.: Modifications of the DIRECT algorithm. Ph.D. thesis, North Carolina State University (2001)
- Gablonsky, J.M., Kelley, C.T.: A locally-biased form of the DIRECT algorithm. Journal of Global Optimization 21(1), 27–37 (2001). DOI 10.1023/A:1017930332101
- Gavana, A.: Global optimization benchmarks and ampgo. http://infinity77.net/ global_optimization/index.html. Online; accessed: 2021-07-22
- Grishagin, V.A.: Operating characteristics of some global search algorithms. In: Problems of Stochastic Search, vol. 7, pp. 198–206. Zinatne, Riga (1978). In Russian
- He, J., Verstak, A., Watson, L.T., Sosonkina, M.: Design and implementation of a massively parallel version of direct. Computational Optimization and Applications (2008). DOI 10.1007/s10589-007-9092-2
- Hedar, A.: Test functions for unconstrained global optimization. http://www-optima. amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestG0.htm (2005). Online; accessed: 2017-03-22

$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	#	Name	Source	n	D	Type	No. of minima	f^*
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	1, 2, 3	$Ackley^{\alpha}$	[15, 54]	2, 5, 10	$[-18, 47]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	4, 5, 6	$Alpine^{\alpha}$	[12]	2, 5, 10	$[\sqrt[i]{2}, 8 + \sqrt[i]{2}]^n$	non-convex	multi-modal	-2.8081^{n}
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	7	Beale	[15,54]	2	$[-4.5, 4.5]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	8	$Bohachevsky1^{\alpha}$	[15, 54]	2	$[-55, 145]^n$	convex	uni-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	9	$Bohachevsky2^{\alpha}$	[15, 54]	2	$[-55, 145]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	10	$Bohachevsky3^{\alpha}$	[15, 54]	2	$[-55, 145]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	11	Booth	[15, 54]	2	$[-10, 10]^n$	convex	uni-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	12	Branin	[15,6]	2	$[-5, 10] \times [10, 15]$	non-convex	multi-modal	0.3978
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	13	Bukin6	[54]	2	$[-15,5] \times [-3,3]$	convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	14	Colville	[15, 54]	4	$[-10, 10]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	15	Cross_in_Tray	[54]	2	$[0, 10]^n$	non-convex	multi-modal	-2.0626
	16	Crossleqtable	[12]	2	$[-10, 15]^n$	non-convex	multi-modal	-1.000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	17.18.19	$Csendes^{\alpha}$	[12]	2, 5, 10	$[-10, 25]^n$	convex	multi-modal	0.0000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	20	Damavandi	12	2	$[0, 14]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	21, 22, 23	$Deb01^{\alpha}$	[12]	2, 5, 10	$[-0.55, 1.45]^n$	non-convex	multi-modal	-1.0000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	24, 25, 26	$Deb02^{\alpha}$	12	2, 5, 10	$[0.225, 1.225]^n$	non-convex	multi-modal	-1.0000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	27.28.29	Dixon_and_Price	[15, 54]	2.5.10	$[-10, 10]^n$	convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	30	Drop wave α	[54]	2, 0, 0	$[-4, 6]^n$	non-convex	multi-modal	-1.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $			[* -]		$[-100^{[-1,0]n}]^n$			
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	31	$Easom^{\alpha}$	[15, 54]	2	$\frac{100}{i+1}$, 100i	non-convex	multi-modal	-1.0000
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	32	Eggholder	[54]	2	$[-512, 512]^n$	non-convex	multi-modal	-959.6406
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	33	$Goldstein_and_Price^{\alpha}$	[15, 6]	2	$[-1.1, 2.9]^n$	non-convex	multi-modal	3.0000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	34, 35, 36	$Griewank^{\alpha}$	[15, 54]	2, 5, 10	$\left[-\sqrt{600i}, \frac{600}{\sqrt{2}}\right]^n$	non-convex	multi-modal	0.0000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $			(1 m m d)					0.000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	37	Hartman3	[15, 54]	3	$[0, 1]^n$	non-convex	multi-modal	-3.8627
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	38	Hartmano	[15,54]	6	[0, 1]"	non-convex	multi-modal	-3.3223
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	39	Holder_Table	[54]	2	$[-10, 10]^n$	non-convex	multi-modal	-19.2085
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	40	Hump	[15, 54]	2	$[-5, 5]^n$	non-convex	multi-modal	-1.0316
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	41	Langermann	[54]	2	$[0, 10]^n$	non-convex	multi-modal	-4.1558
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	42, 43, 44	Levy	[15, 54]	2, 5, 10	$[-10, 10]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{c c c c c c c c c c c c c c c c c c c $	45	$Matyas^{\alpha}$	[15, 54]	2	$[-5.5, 14.5]^n$	convex	uni-modal	0.0000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	46	McCormick	[54]	2	$[-1.5, 4] \times [-3, 4]$	convex	multi-modal	-1.9132
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	47	Michalewicz	[15, 54]	2	$[0, \pi]^n$	non-convex	multi-modal	-1.8013
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	48	Michalewicz	[15, 54]	5	$[0, \pi]^n$	non-convex	multi-modal	-4.6876
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	49	Michalewicz	[15, 54]	10	$[0, \pi]^n$	non-convex	multi-modal	-9.6601
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	50	Perm4	[15, 54]	4	$[-i, i]^n$	non-convex	multi-modal	0.0000
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	51, 52, 53	$Pinter^{\alpha}$	[12]	2, 5, 10	$[-5.5, 14.5]^n$	non-convex	multi-modal	0.0000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	54	Powell	[15, 54]	4	$[-4, 5]^n$	convex	multi-modal	0.0000
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	55	$Power_Sum^{\alpha}$	[15, 54]	4	$[1, 4 + \sqrt[n]{2}]^n$	convex	multi-modal	0.0000
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	56, 57, 58	Qing	[12]	2, 5, 10	$[-500, 500]^n$	non-convex	multi-modal	0.0000
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	59, 60, 61	$Rastrigin^{\alpha}$	[15, 54]	2, 5, 10	$[-5\sqrt[4]{2}, 7 + \sqrt[4]{2}]_n^n$	non-convex	multi-modal	0.0000
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	62, 63, 64	$Rosenbrock^{\alpha}$	[15, 6]	2, 5, 10	$\left -\frac{5}{\sqrt{i}}, 10\sqrt{i}\right ^{n}$	non-convex	uni-modal	0.0000
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	65,66,67	$Rotated_H_Ellip^{\alpha}$	[54]	2, 5, 10	[-35,96] ⁿ	convex	uni-modal	0.0000
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	68, 69, 70	$Schwefel^{\alpha}$	[15, 54]	2,5,10	$\left -500 + \frac{100}{\sqrt{i}}, 500 - \frac{40}{\sqrt{i}} \right ^n$	non-convex	multi-modal	0.0000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	71	Shekel5	[15, 54]	4	$[0, 10]^n$	non-convex	multi-modal	-10.1531
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	72	Shekel7	[15, 54]	4	$[0, 10]^n$	non-convex	multi-modal	-10.4029
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	73	Shekel10	[15, 54]	4	$[0, 10]^n$	non-convex	multi-modal	-10.5364
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	74	Shubert	[15,54]	2	$[-10, 10]^n$	non-convex	multi-modal	-186.7309
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	75, 76, 77	$Sphere^{\alpha}$	[15, 54]	2, 5, 10	$[-2.75, 7.25]^n$	convex	uni-modal	0.0000
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	78, 79, 80	$Styblinski_Tana^{\alpha}$	[4]	2, 5, 10	$[-5, 5 + \sqrt[i]{3}]^n$	non-convex	multi-modal	-39.1661n
$ \begin{array}{c ccccccccccccccccccccccccccccccccccc$	81, 82, 83	$Sum_of_Powers^{\alpha}$	[54]	2, 5, 10 2, 5, 10	$[-0.55, 1, 45]^n$	convex	uni-modal	0.0000
$\begin{array}{c c c c c c c c c c c c c c c c c c c $	84 85 86	Sum Square ^a	[/]	2, 5, 10 2 5 10	$[-5, 5, 14, 5]^n$	convex	uni-modal	0.0000
$\begin{array}{c ccccccccccccccccccccccccccccccccccc$	87	Trefethen	[12]	2, 3, 10	[-2 2]n	non-convex	multi-modal	-3.3068
$ \begin{array}{cccccccccccccccccccccccccccccccccccc$	88 89 90	Trid	[15 54]	2 5 10	$\begin{bmatrix} 2, 2 \end{bmatrix}$	convey	multi-modal	3
$\begin{array}{cccccccccccccccccccccccccccccccccccc$	91, 92, 93	Vincent	[10,04]	2, 5, 10 2, 5, 10	$[0.25 \ 10]^n$	non-convex	multi-modal	_n
	94 95 96	Zakharov ^α	[15 54]	2, 5, 10 2 5 10	$[-1.625, 13.375]^n$	convey	multi-modal	0 0000
		2	[10,04]	2, 0, 10	[1.020, 10.010]	convex	inutr-modal	0.0000

Table 5 Key characteristics of the DIRECTGOLib v1.1[51,47] test problems for box-constrained global optimization

 $\vartheta - -\frac{1}{6}n^3 - \frac{1}{2}n^2 + \frac{2}{3}n$ α - domain *D* was taken from [45] i = 1, ..., n

- 16. Holmstrom, K., Goran, A.O., Edvall, M.M.: User's guide for tomlab 7 (2010). URL https: //tomopt.com/
- 17. Horst, R., Pardalos, P.M., Thoai, N.V.: Introduction to Global Optimization. Nonconvex Optimization and Its Application. Kluwer Academic Publishers, Berlin, Germany (1995)
- 18. Jones, D.R.: The DIRECT global optimization algorithm. In: C.A. Floudas, P.M. Pardalos (eds.) The Encyclopedia of Optimization, pp. 431-440. Kluwer Academic Publishers, Dordrect (2001)
- 19. Jones, D.R., Martins, J.R.R.A.: The DIRECT algorithm: 25 years later. Journal of Global Optimization 79, 521-566 (2021). DOI 10.1007/s10898-020-00952-6

- Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. Journal of Optimization Theory and Application 79(1), 157–181 (1993). DOI 10.1007/BF00941892
- Liberti, L., Kucherenko, S.: Comparison of deterministic and stochastic approaches to global optimization. International Transactions in Operational Research 12(3), 263-285 (2005). DOI https://doi.org/10.1111/j.1475-3995.2005.00503.x. URL https:// onlinelibrary.wiley.com/doi/abs/10.1111/j.1475-3995.2005.00503.x
- Liu, H., Xu, S., Chen, X., Wang, X., Ma, Q.: Constrained global optimization via a directtype constraint-handling technique and an adaptive metamodeling strategy. Structural and Multidisciplinary Optimization 55(1), 155–177 (2017). DOI 10.1007/s00158-016-1482-6
- Liuzzi, G., Lucidi, S., Piccialli, V.: A DIRECT-based approach exploiting local minimizations for the solution for large-scale global optimization problems. Computational Optimization and Applications 45(2), 353–375 (2010). DOI 10.1007/s10589-008-9217-2
- Liuzzi, G., Lucidi, S., Piccialli, V.: Exploiting derivative-free local searches in DIRECT-type algorithms for global optimization. Computational Optimization and Applications 65, 449–475 (2016). DOI DOI10.1007/s10589-015-9741-9
- Na, J., Lim, Y., Han, C.: A modified direct algorithm for hidden constraints in an lng process optimization. Energy 126, 488-500 (2017). DOI https://doi.org/ 10.1016/j.energy.2017.03.047. URL https://www.sciencedirect.com/science/article/ pii/S0360544217304164
- Paulavičius, R., Chiter, L., Žilinskas, J.: Global optimization based on bisection of rectangles, function values at diagonals, and a set of Lipschitz constants. Journal of Global Optimization 71(1), 5–20 (2018). DOI 10.1007/s10898-016-0485-6
 Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased DISIMPL
- Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. Journal of Global Optimization 59(2-3), 545–567 (2014). DOI 10.1007/s10898-014-0180-4
- Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased BIRECT algorithm with local accelerators for expensive global optimization. Expert Systems with Applications 144, 11305 (2020). DOI 10.1016/j.eswa.2019.113052
- Paulavičius, R., Žilinskas, J.: Analysis of different norms and corresponding Lipschitz constants for global optimization. Technological and Economic Development of Economy 36(4), 383–387 (2006). DOI 10.1080/13928619.2006.9637758
- Paulavičius, R., Žilinskas, J.: Analysis of different norms and corresponding Lipschitz constants for global optimization in multidimensional case. Information Technology and Control 36(4), 383–387 (2007)
- Paulavičius, R., Žilinskas, J.: Simplicial Lipschitz optimization without the Lipschitz constant. Journal of Global Optimization 59(1), 23–40 (2014). DOI 10.1007/s10898-013-0089-3.
- Paulavičius, R., Žilinskas, J.: Advantages of simplicial partitioning for Lipschitz optimization problems with linear constraints. Optimization Letters 10(2), 237–246 (2016). DOI 10.1007/s11590-014-0772-4
- Pillo, G.D., Liuzzi, G., Lucidi, S., Piccialli, V., Rinaldi, F.: A DIRECT-type approach for derivative-free constrained global optimization. Computational Optimization and Applications 65(2), 361–397 (2016). DOI 10.1007/s10589-016-9876-3
- Pillo, G.D., Lucidi, S., Rinaldi, F.: An approach to constrained global optimization based on exact penalty functions. Journal of Optimization Theory and Applications 54(2), 251– 260 (2010). DOI 10.1007/s10898-010-9582-0
- Pinter, J.D.: Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications, Nonconvex Optimization and Its Applications, vol. 6. Springer US, Berlin, Germany (1996). DOI 10.1007/ 978-1-4757-2502-5
- Piyavskii, S.A.: An algorithm for finding the absolute minimum of a function. Theory of Optimal Solutions 2, 13–24 (1967). DOI 10.1016/0041-5553(72)90115-2. In Russian
- Rios, L.M., Sahinidis, N.V.: Derivative-free optimization: a review of algorithms and comparison of software implementations. Journal of Global Optimization 56(3), 1247– 1293 (2013). DOI 10.1007/s10898-012-9951-y
- Sergeyev, Y.D., Kvasov, D., Mukhametzhanov, M.: On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Scientific reports 8(1), 1–9 (2018). DOI 10.1038/s41598-017-18940-4
- Sergeyev, Y.D., Kvasov, D.E.: Global search based on diagonal partitions and a set of Lipschitz constants. SIAM Journal on Optimization 16(3), 910–937 (2006). DOI 10.1137/ 040621132

- Sergeyev, Y.D., Kvasov, D.E.: Diagonal Global Optimization Methods. FizMatLit, Moscow (2008). In Russian
- 41. Sergeyev, Y.D., Kvasov, D.E.: Lipschitz global optimization. In: J.J. Cochran, L.A. Cox, P. Keskinocak, J.P. Kharoufeh, J.C. Smith (eds.) Wiley Encyclopedia of Operations Research and Management Science (in 8 volumes), vol. 4, pp. 2812–2828. John Wiley & Sons, New York, NY, USA (2011)
- Sergeyev, Y.D., Kvasov, D.E.: Deterministic Global Optimization: An Introduction to the Diagonal Approach. SpringerBriefs in Optimization. Springer, Berlin, Germany (2017). DOI 10.1007/978-1-4939-7199-2
- Shubert, B.O.: A sequential method seeking the global maximum of a function. SIAM Journal on Numerical Analysis 9, 379–388 (1972). DOI 10.1137/0709036
- 44. Stripinis, L., Paulavičius, R.: A new DIRECT-GLh algorithm for global optimization with hidden constraints. Optimization Letters 15(6), 1865–1884 (2021). DOI 10.1007/ s11590-021-01726-z. URL https://doi.org/10.1007/s11590-021-01726-z
- 45. Stripinis, L., Paulavičius, R.: An empirical study of various candidate selection and partitioning techniques in the DIRECT framework. Journal of Global Optimization (2022). DOI 10.1007/s10898-022-01185-5. URL https://doi.org/10.1007/s10898-022-01185-5
- 46. Stripinis, L., Paulavičius, R.: Directgo: A new direct-type matlab toolbox for derivativefree global optimization, version v1.1.0, *GitHub*. https://github.com/blockchain-group/ DIRECTGO/releases/tag/v1.1.0 (2022)
- Stripinis, L., Paulavičius, R.: DIRECTGOLib DIRECT Global Optimization test problems Library, Version v1.1, GitHub. https://github.com/blockchain-group/ DIRECTGOLib/tree/v1.1 (2022)
- Stripinis, L., Paulavičius, R., Žilinskas, J.: Improved scheme for selection of potentially optimal hyper-rectangles in DIRECT. Optimization Letters 12(7), 1699–1712 (2018). DOI 10.1007/s11590-017-1228-4
- Stripinis, L., Paulavičius, R., Žilinskas, J.: Penalty functions and two-step selection procedure based DIRECT-type algorithm for constrained global optimization. Structural and Multidisciplinary Optimization 59(6), 2155–2175 (2019). DOI 10.1007/s00158-018-2181-2
- Stripinis, L., Paulavičius, R.: DIRECTGO: A New DIRECT-Type MATLAB Toolbox for Derivative-Free Global Optimization. ACM Transactions on Mathematical Software (2022). DOI 10.1145/3559755. URL https://doi.org/10.1145/3559755
- Stripinis, L., Paulavičius, R.: DIRECTGOLib DIRECT Global Optimization test problems Library, Version v1.1, Zenodo (2022). DOI 10.5281/zenodo.6491951. URL https://doi.org/10.5281/zenodo.6491951
- 52. Stripinis, L., Žilinskas, J., Casado, L.G., Paulavičius, R.: On matlab experience in accelerating direct-glce algorithm for constrained global optimization through dynamic data structures and parallelization. Applied Mathematics and Computation 390, 1–17 (2021). DOI https://doi.org/10.1016/j.amc.2020.125596. URL https://www.sciencedirect.com/science/article/pii/S0096300320305518
- 53. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-Convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic Publishers, Dordrecht (2000)
- 54. Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: Test functions and datasets. http://www.sfu.ca/~ssurjano/index.html (2013). Online; accessed: 2017-03-22