

# GENDIRECT: a GENERALized DIRECT-type algorithmic framework for derivative-free global optimization

Linus Stripinis · Remigijus Paulavičius

Received: date / Accepted: date

**Abstract** Over the past three decades, numerous articles have been published discussing the renowned **DIRECT** algorithm (DIdiding RECTangles). These articles present innovative ideas to enhance its performance and adapt it to various types of optimization problems. To consolidate and summarize this progress, we have recently introduced **DIRECTGO**—a comprehensive collection featuring more than fifty deterministic, derivative-free algorithmic implementations based on the **DIRECT** framework. **DIRECTGO** empowers users to conveniently employ diverse **DIRECT**-type algorithms, enabling efficient solutions to practical optimization problems. Despite their variations, **DIRECT**-type algorithms share a common algorithmic structure and typically differ only at certain steps.

Recognizing this, we take further steps in generalization within this paper and propose **GENDIRECT**—GENERALized **DIRECT**-type framework that encompasses and unifies **DIRECT**-type algorithms under a single generalized approach. **GENDIRECT** offers a practical alternative to the creation of yet another “new” **DIRECT**-type algorithm that closely resembles existing ones. Instead, **GENDIRECT** allows the efficient generation of known or novel **DIRECT**-type optimization algorithms by assembling different algorithmic components. This approach provides considerably more flexibility compared to both the **DIRECTGO** toolbox and individual **DIRECT**-type algorithms. In general, **GENDIRECT** allows the creation of approximately a few hundred thousand combinations of **DIRECT**-type algorithms, facilitating user-friendly customization and the incorporation of new algorithmic components for further advancements.

By modifying specific components of five highly promising **DIRECT**-type algorithms found in the existing literature using **GENDIRECT**, the significant potential of **GENDIRECT** has been demonstrated. The resulting newly developed

---

L. Stripinis · R. Paulavičius  
Institute of Data Science and Digital Technologies, Vilnius University, Akademijos 4, LT-08663, Vilnius, Lithuania  
E-mail: linas.stripinis@mif.vu.lt  
R. Paulavičius  
E-mail: remigijus.paulavicius@mif.vu.lt

improved approaches exhibit greater efficiency and enhanced robustness in dealing with problems of varying complexity.

**Keywords** Derivative-free global optimization · DIRECT-type algorithms · Optimization software · Numerical benchmarking

**Mathematics Subject Classification (2020)** 90C26 · 65K10

## 1 Introduction

Optimization problems encountered in scientific and engineering domains often involve objective functions that can only be obtained through “black-box” methods or simulations, lacking derivative information. For example, Google’s internal services frequently employ black-box optimization techniques with automated parameter tuning engines [10]. Furthermore, objective function evaluations are becoming more computationally expensive as applications grow in size and complexity [22]. Consequently, calculating derivatives is often infeasible or impractical. As a result, there is a growing emphasis on the development of derivative-free global optimization (DFGO) methods. These methods are specifically designed to address the growing complexity and diversity of optimization problems, where derivative information is neither available nor practical to compute. This active development of DFGO methods addresses the need for efficient optimization techniques in scenarios where derivatives cannot be utilized.

This paper considers a box-constrained single-objective optimization problem

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

where  $f : \mathbb{R}^n \rightarrow \mathbb{R}$  is a potentially “black-box” Lipschitz-continuous objective function with an unknown Lipschitz constant, and  $\mathbf{x} \in \mathbb{R}^n$  is the input vector of control variables. Moreover,  $f$  can be non-linear, multi-modal, non-convex, and non-differentiable. We assume that  $f$  can only be computed at any point of the feasible region, which is a  $n$ -dimensional hyper-rectangle

$$D = [\mathbf{a}, \mathbf{b}] = \{\mathbf{x} \in \mathbb{R}^n : a_j \leq x_j \leq b_j, j = 1, \dots, n\}.$$

However, there is no access to additional information on the objective function  $f(\mathbf{x})$ , such as gradients and the Hessian, as is typical for a “black-box” case.

Among the solution techniques available for a given problem (1), population-based meta-heuristic methods have gained widespread popularity. Numerous approaches have been proposed and developed within this category [2]. For global optimization problems that involve costly evaluations, model-based optimization algorithms are commonly employed. Among these algorithms, Bayesian optimization [18] and various surrogate models [21] stand out as the leading state-of-the-art methods for optimizing expensive “black-box” functions.

DIRECT [17] presents an alternative specifically tailored for “black-box” global optimization by extending the classical Lipschitz optimization [36, 38, 39, 41], eliminating the requirement of knowing the Lipschitz constant. In contrast to the stochastic methods discussed above, the DIRECT-type algorithms adhere to a

deterministic pattern. A recent comprehensive numerical benchmark study involving various derivative-free global optimization solvers [45] highlighted that particularly for problems with lower dimensions, DIRECT-type algorithms can significantly outperform stochastic approaches. Furthermore, certain combinations of hybrid local search algorithms based on DIRECT-type methods and finite differences [42] demonstrated exceptional efficiency in solving high-dimensional problems. Consequently, designing and developing efficient DIRECT-type algorithms is crucial and driven by practical needs.

Inspired by these observations, we have recently introduced **DIRECTGO**, a **MATLAB** toolbox dedicated to DFGO. The latest release of **DIRECTGO** includes a comprehensive collection of 52 distinct algorithmic implementations based on the **DIRECT** framework. However, recent empirical studies [50, 51] have highlighted that even more efficient DIRECT-type algorithms can be achieved by innovatively combining existing algorithmic steps. It seems that many authors may not spend enough time exploring the most suitable algorithmic framework when developing and publishing new algorithms of type **DIRECT**.

Therefore, this study introduces a novel framework called **GENDIRECT**, which offers a **GEN**eralized **DIRECT**-type approach to derivative-free global optimization. **GENDIRECT** enables the construction of any known or previously unexplored DIRECT-type algorithm. Instead of developing yet another “new” DIRECT-type algorithm, **GENDIRECT** provides a rapid and effective way of combining different components to create customized DIRECT-type algorithms.

Using **GENDIRECT**, users can identify and utilize the most suitable DIRECT-type algorithm for a given optimization problem based on the latest advances in the field. Compared to the **DIRECTGO** toolbox and individual DIRECT-type algorithms, the **GENDIRECT** framework offers a significantly higher level of flexibility. In fact, **GENDIRECT** allows the design of a few hundred thousand combinations of DIRECT-type algorithms and facilitates user-friendly experimentation with new algorithmic components.

**GENDIRECT** is implemented as a separate extension of **DIRECTGO**, complemented by a dedicated graphical user interface (GUI). This GUI provides easy access to all the features and capabilities of **GENDIRECT**, ensuring a seamless user experience.

The capability of **GENDIRECT** is showcased by selecting five highly promising DIRECT-type algorithms from the existing literature, as identified in [45, 52]. By leveraging **GENDIRECT**, specific components that were identified as weaknesses in these algorithms are modified. As a result, some of these algorithms demonstrate significantly improved efficiency, showcasing the potential of **GENDIRECT** in optimizing and refining DIRECT-type algorithms using the most recent **DIRECTGOLib v2.0**.

This work makes several significant contributions, including

1. Introduction of a novel framework called **GENDIRECT**, which represents a **GEN**eralized **DIRECT**-type algorithmic framework.
2. **GENDIRECT** provides an efficient and innovative approach to generate DIRECT-type optimization algorithms, whether they are existing algorithms or entirely novel ones, by combining different algorithmic components.
3. **GENDIRECT** allows for the creation of a few hundred thousand combinations of DIRECT-type algorithms, facilitating user-friendly experimentation and enabling new developments in optimization.

4. Description of the implementation of **GENDIRECT** as an evolution of **DIRECTGO**, complete with a separate graphical user interface (GUI) that ensures easy access to all its features. This implementation is free and open for anyone to use.
5. Demonstration of the potential of **GENDIRECT** by enhancing the efficiency of five chosen **DIRECT**-type algorithms through modifications. These modifications showcase the ability of **GENDIRECT** to improve algorithmic performance further.

In summary, this work contributes to the derivative-free global optimization field by introducing **GENDIRECT**, a versatile framework that enables efficient algorithm generation, offers extensive customization options, and shows improved efficiency in established **DIRECT**-type algorithms.

The remainder of the paper is organized as follows. Section 2 presents a concise overview of key advancements in the realm of **DIRECT**-type algorithms. Section 3 introduces and elaborates on the **GENDIRECT** framework. The experimental results of the newly developed algorithms and performance evaluation utilizing **GENDIRECT** are analyzed in Section 4. Lastly, Section 5 offers concluding remarks and outlines potential avenues for future exploration in this field.

## 2 Background for **GENDIRECT**

### 2.1 General structure of **DIRECT**-type algorithms

The **DIRECT** algorithm was originally designed to solve global optimization problems with box constraints (1). Despite numerous proposals, most follow a similar algorithmic structure and involve three primary steps: selection, sampling, and partitioning (see Algorithm 1). However, at first, **DIRECT**-type algorithms typically transform a feasible region  $D = [\mathbf{a}, \mathbf{b}]$  into a unit hyper-rectangle  $\bar{D} = [0, 1]^n$  referring to the original space ( $D$ ) solely to evaluate the objective function  $f$  (as depicted in Algorithm 1, Lines 1–5).

The selection, partitioning, and sampling operations are executed within a normalized search domain  $\bar{D}$ . During each iteration, specific regions are identified as potentially optimal candidates (POC) and chosen for further investigation (see Algorithm 1, Line 7). In **DIRECT**-type algorithms, the objective function is sampled and evaluated at various points within each POC, which are then subdivided into smaller sub-regions (see Algorithm 1, Lines 9 and 10). This selection, sampling, and subdivision process continues until a predefined limit is reached.

The subsequent subsections provide an overview of the primary techniques proposed for each step. Although the selection step precedes sampling and partitioning, we will initially focus on the latter because the selection step relies directly on the strategies employed in sampling and partitioning.

### 2.2 Summary of sampling and partitioning schemes

In this section, we present a brief summary of seven primary sampling and partitioning approaches that have been proposed in existing literature [16, 17, 33, 37, 40, 52] and implemented within the **GENDIRECT** framework. Table 1

**Algorithm 1:** Main steps of DIRECT-type algorithms

---

**input** : Objective function ( $f$ ), search domain ( $D$ ), and adjustable algorithmic options ( $opt$ ): goal for the function value ( $f^{\text{goal}}$ ), maximal number of function evaluations ( $M_{\text{max}}$ ) and algorithmic iterations ( $K_{\text{max}}$ ) ;

**output**: The best found objective value ( $f^{\text{min}}$ ), solution point ( $\mathbf{x}^{\text{min}}$ ), and record of various performance metrics: percent error ( $pe$ ), number of iterations ( $k$ ), number of function evaluations ( $m$ );

---

**Initialization step:**

- 1 *Normalize* the search domain  $D$  to the unit hyper-rectangle  $\bar{D}$ ;
- 2 *Evaluate*  $f$  at initial sampling point(s) and set:
- 3  $x_j^{\text{min}} = \lfloor b_j - a_j \rfloor \bar{x}_j + a_j, j = 1, \dots, n;$  // referring to  $D$
- 4  $f^{\text{min}} = f(\mathbf{x}^{\text{min}});$
- 5 *Initialize* performance measures:  $k = 1, m = 1$  ;
- 6 **while**  $f^{\text{goal}} < f^{\text{min}}$  **and**  $m < M_{\text{max}}$  **and**  $k < K_{\text{max}}$  **do**
- 7     **Selection step:** *Identify* the set  $S_k$  of POCs;
- 8     **foreach**  $\bar{D}_k^j \in S_k$  **do**
- 9         **Sampling step:** *Evaluate*  $f$  at newly sampled points in  $\bar{D}_k^j$ ;
- 10        **Partitioning step:** *Subdivide*  $\bar{D}_k^j$  ;
- 11     **end**
- 12     **if Hybrid then**
- 13         *Run* local search procedure; // only in hybrid version
- 14     **end**
- 15     *Update*  $f^{\text{min}}, \mathbf{x}^{\text{min}}$ , and performance measures:  $k$  and  $m$ ;
- 16 **end**
- 17 **Return**  $f^{\text{min}}, \mathbf{x}^{\text{min}}$ , and performance measures  $(k, m)$ .

---

provides an overview of these schemes, including illustrative examples from the initial iterations. Blue-colored sub-regions indicate the POCs in the current partition.

Although each of the seven schemes possesses distinct characteristics, they demonstrate significant similarities. Particularly, these schemes involve sampling new points and subdividing larger regions into smaller, non-overlapping sub-regions. In cases where there is more than one longest side, two primary strategies for division emerge:

- Subdivision along all dimensions with the maximum side length.
- Subdivision along a single dimension with maximum side length.

It is worth mentioning that the original DIRECT algorithm proposed subdividing along all dimensions. However, extensive experimentation has indicated that this approach does not consistently yield effective results.

**Table 1** Summary of sampling and partitioning schemes commonly utilized in **DIRECT**-type algorithms implemented within **GENDIRECT** (in ascending order of the year of publication)

Notation & Source	Partitioning and sampling scheme	An example of the initialization and two subsequent iterations
<b>DTC</b> [16]	A hyper-rectangular partition based on one- <b>D</b> imensional <b>T</b> rissection, and sampling points located at <b>C</b> enters.	
<b>DTDV</b> [40]	A hyper-rectangular partition based on one- <b>D</b> imensional <b>T</b> rissection, and sampling points located at two <b>D</b> iagonal <b>V</b> ertices.	
<b>DTCS</b> [37]	A simplicial partition based on one- <b>D</b> imensional <b>T</b> rissection, and sampling points located at <b>C</b> enters of <b>S</b> implices.	
<b>DBVS</b> [37]	A simplicial partition based on one- <b>D</b> imensional <b>B</b> isection, and sampling points located at <b>V</b> ertices of <b>S</b> implices.	
<b>DBDP</b> [33]	A hyper-rectangular partition based on one- <b>D</b> imensional <b>B</b> isection, and sampling points located at two <b>D</b> iagonal <b>P</b> oints equidistant between themselves and a diagonal's vertices.	
<b>DBVD</b> [4]	A hyper-rectangular partition based on one- <b>D</b> imensional <b>B</b> isection, and sampling points located at one <b>V</b> ertice and one <b>D</b> iagonal point with a 2:3 diagonal distance from the sampling vertice.	
<b>DBC</b> [52]	A hyper-rectangular partition based on one- <b>D</b> imensional <b>B</b> isection, and sampling points located at <b>C</b> enter points.	

### 2.3 Selection schemes

Initially, selecting POCs is straightforward since only one candidate is available, the entire feasible region. However, to introduce selection schemes in subsequent iterations, we must first establish the concept of the current partition ( $\mathcal{P}_k$ ), which in iteration  $k$ , is defined as

$$\mathcal{P}_k = \{\bar{D}_k^i : i \in \mathbb{I}_k\},$$

where  $\bar{D}_k^i$  are hyper-rectangles (or simplices) and  $\mathbb{I}_k$  is the index set identifying the current partition  $\mathcal{P}_k$ . Then, the next partition,  $\mathcal{P}_{k+1}$ , is obtained by subdividing the selected POCs from the current partition  $\mathcal{P}_k$ .

When identifying POCs, two crucial aspects come into play: the  $\bar{D}_k^i$  measure ( $\bar{\delta}_k^i$ ) and the general quality based on the function values attained at the sample points.

### 2.3.1 Evaluating goodness of candidates

The values of the objective function obtained from the sampled points  $\mathbb{H}_k^i$  are utilized to assess the overall quality of the candidate. We refer to this value as the aggregated function value ( $\mathcal{F}_k^i$ ), which represents the goodness of  $\bar{D}_k^i$ . In summary, four strategies have been presented to evaluate  $\mathcal{F}_k^i$  in the literature [52] as defined in Definition 1.

**Definition 1** (Aggregated function values) Let:

- $\bar{\delta}_k^i$  is a measure of  $\bar{D}_k^i$ ;
- $\mathbf{x}_m^i$  is a midpoint of  $\bar{D}_k^i$ ;
- $\mathbf{x}^{\min}$  is a currently best found minimum point;
- $\mathbb{H}_k^i$  is a representative sampling index set of all sample points within  $\bar{D}_k^i$ ;
- $\text{card}(\mathbb{H}_k^i)$  is the cardinality of a set  $\mathbb{H}_k^i$ .

Then:

- *Midpoint value based aggregated function value:*

$$\mathcal{F}_k^i = f(\mathbf{x}_m^i), \quad (2)$$

- *Minimum value based aggregated function value:*

$$\mathcal{F}_k^i = \min_{j \in \mathbb{H}_k^i} f(\mathbf{x}^j) \quad (3)$$

- *Mean value based aggregated function value:*

$$\mathcal{F}_k^i = \frac{1}{\text{card}(\mathbb{H}_k^i)} \sum_{j=1}^{\text{card}(\mathbb{H}_k^i)} f(\mathbf{x}^j) \quad (4)$$

- *Midpoint and minimum values based aggregated function value:*

$$\mathcal{F}_k^i = \frac{1}{2} \left( \min_{j \in \mathbb{H}_k^i} f(\mathbf{x}^j) + f(\mathbf{x}_m^i) \right) \quad (5)$$

The use of the  $\mathcal{F}_k^i$  evaluation strategy depends on the specific sampling strategy being utilized. For example, in the case of the DTC and DTCS schemes, the midpoint value-based  $\mathcal{F}_k^i$  is adopted since sampling is performed solely at a single midpoint. However, when there are multiple sampling points per candidate, alternative strategies have been shown to have a significant influence, as shown in previous work [47].

### 2.3.2 Measuring candidates

Depending on the sampling strategy, basically only two different ways have been proposed to measure POCs:

$$\bar{\delta}_k^i = \lambda d_k^i, \quad (6)$$

$$\bar{\delta}_k^i = \max_{j, l \in \mathbb{H}_k^i} \|\mathbf{x}^j - \mathbf{x}^l\|_2. \quad (7)$$

where  $\lambda \in [0, 1]$  and  $d_k^i$  represents the Euclidean length of the diagonal of  $\bar{D}_k^i$  diagonal. A couple of significant points should be emphasized in this regard. First, instead of relying solely on the Euclidean norm, alternative norms (e.g.,  $\|\cdot\|_\infty$ ) have been observed to yield favorable results, as noted in the work [8]. Second, different partitioning schemes have employed various values for  $\lambda$ . For example, some schemes use  $\lambda = 1$ , which corresponds to the full length of the diagonal, as seen in [52], while others adopt  $\lambda = 2/3$ , as demonstrated in [33]. However, since  $\lambda$  applies uniformly to all partition elements  $\bar{D}_k^i$  and serves solely to counterbalance the selection of POC, the choice of  $\lambda$  does not affect the performance of algorithms of type **DIRECT**.

### 2.3.3 Summary of POC selection schemes

To address the identified limitations of **DIRECT**-type algorithms, various POC selection schemes have been proposed. Definition 2 defines the four most widely used and implemented selection schemes in **GENDIRECT**, while a summary of them is given in Table 1.

**Definition 2** (Selection schemes) Let:

- $\mathcal{F}_k^i$  denotes the aggregated function value for  $\bar{D}_k^i$ ;
- $\bar{\delta}_k^i$  is a measure of  $\bar{D}_k^i$ ;
- $\mathbb{I}_k^i \subseteq \mathbb{I}_k$  represents a subset of indices that correspond to elements of  $\mathcal{P}_k$  sharing the same measure ( $\bar{\delta}_k^i$ ). Additionally,  $\mathbb{I}_k^{\min}$  contains the indices of elements with the smallest measure,  $\bar{\delta}_k^{\min}$ , while  $\mathbb{I}_k^{\max}$  — with the largest.

Then:

- *Original selection*: A candidate  $\bar{D}_k^j, j \in \mathbb{I}_k$  is said to be potentially optimal if there exists some rate-of-change (Lipschitz) constant  $\tilde{L} > 0$  such that

$$\mathcal{F}_k^j - \tilde{L}\bar{\delta}_k^j \leq \mathcal{F}_k^i - \tilde{L}\bar{\delta}_k^i, \quad \forall i \in \mathbb{I}_k, \quad (8)$$

- *Aggressive selection*: For each  $\mathbb{I}_k^i$  ( $\min \leq i \leq \max$ ) select  $\bar{D}_k^j, j \in \mathbb{I}_k^i$  with the lowest  $\mathcal{F}_k^j$ , i.e.,

$$\mathcal{F}_k^j \leq \mathcal{F}_k^l, \quad \forall l \in \mathbb{I}_k^i. \quad (9)$$

- *Pareto selection*: Select all candidates  $\bar{D}_k^i, i \in \mathbb{I}_k$  who are not dominated, which means that there is no other candidate  $\bar{D}_k^j, j \in \mathbb{I}_k$  that satisfies the condition:

$$(\delta_k^j \geq \delta_k^i \wedge \mathcal{F}_k^j < \mathcal{F}_k^i) \vee (\delta_k^j > \delta_k^i \wedge \mathcal{F}_k^j \leq \mathcal{F}_k^i). \quad (10)$$



- *Reduced Pareto selection*: Select  $\bar{D}_k^i$  with the lowest  $\mathcal{F}_k^i$  and  $\bar{D}_k^j$  with the most extensive measure  $\delta_k^j$ , breaking ties in favor of a lower value of the aggregate function.

In summary, aggressive selection aims to choose a comprehensive set of candidates, ensuring that at least one candidate is selected from each group with different diameters ( $\delta_k^i$ ) while prioritizing candidates with the lowest aggregated function value. Then, the number of candidates selected through Pareto-based criteria tends to exceed the original selection strategy. However, this approach, which emphasizes exploring candidates of intermediate sizes, can lead to slower convergence, particularly when dealing with less complex optimization problems. Therefore, the primary motivation behind introducing a reduced set of Pareto-optimal candidates was to address this issue.

It is important to note that when multiple equally good POC exist with the same  $\delta_k^i$  and  $\mathcal{F}_k^i$ , two distinct selection strategies are available:

- Select all equally good POC;
- Select only one with the highest index number.

Furthermore, selection schemes can integrate additional conditions to enhance the balance between local and global directions. The subsequent subsection provides a detailed examination of these conditions.

#### 2.3.4 Additional approaches for improved local and global POC selection.

**Excessive local refinement reduction techniques.** To protect the algorithm against excessive refinement around current local minima  $f^{\min}$ , the authors in the DIRECT literature [6, 17, 25] proposed incorporating one of the following conditions along with Eq. (8) in the original selection scheme:

$$\mathcal{F}_k^j - \tilde{L}\delta_k^j \leq f^{\min} - \varepsilon|f^{\min}|, \quad (11)$$

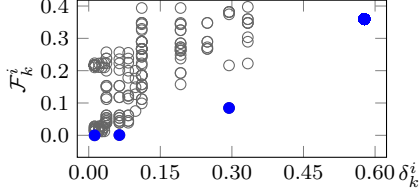
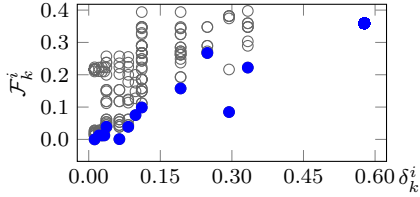
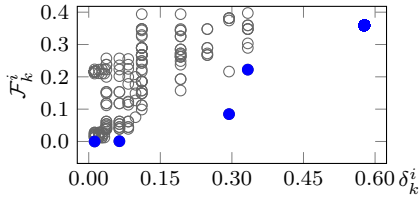
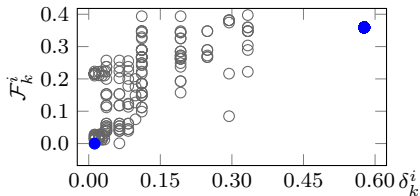
$$\mathcal{F}_k^j - \tilde{L}\delta_k^j \leq f^{\min} - \varepsilon|f^{\min} - f^{\text{median}}|, \quad (12)$$

$$\mathcal{F}_k^j - \tilde{L}\delta_k^j \leq f^{\min} - \varepsilon|f^{\min} - f^{\text{average}}|. \quad (13)$$

Therefore, the lower Lipschitz bound of the POC must be lower than the current minimum value ( $f^{\min}$ ) to at least some extent. The parameter  $\varepsilon$  plays a crucial role in determining the adjustment of the lower Lipschitz bound. In the study conducted by [17], favorable results were achieved using values of  $\varepsilon$  ranging from  $10^{-3}$  to  $10^{-7}$ , and a default value of  $\varepsilon = 10^{-4}$  is suggested. To reduce the sensitivity of the objective function to additive scaling, subtraction of the median value ( $f^{\text{median}}$ ) or the average ( $f^{\text{average}}$ ) value (as shown in Eqs. (12) and (13)) was proposed.

**Restart technique for the  $\varepsilon$  parameter.** In [5], an adaptive scheme is introduced for the parameter  $\varepsilon$  to prevent wasteful function evaluations in minor regions  $\bar{D}_k^i$  where negligible improvements are expected. The restart technique begins with  $\varepsilon = 0$  and is maintained until an improvement is observed. However, if there is no improvement for five consecutive iterations, it suggests a potential stagnation at a local optimum. To address this, the algorithm switches to  $\varepsilon = 0.01$ . Within 50 iterations, the restart technique returns to  $\varepsilon = 0$  if an improvement is found or no progress is made. If another 50 iterations pass

**Table 2** Summary of selection schemes implemented in GENDIRECT

Notation & Source	Description	Illustration of POCs selection (blue points) using DIRECT on the same sample
<b>Original</b> [17]	<i>Original selection strategy.</i> Selects POCs based on the lower Lipschitz bound estimates for all possible Lipschitz constant values.	
<b>Aggressive</b> [3]	<i>Aggressive selection strategy.</i> Selects at least one candidate from each group of different diameters.	
<b>Pareto</b> [29]	<i>Pareto selection strategy.</i> Selects all candidates that are non-dominated on size (the higher, the better) and aggregated function value (the lower, the better).	
<b>Reduced Pareto</b> [30]	<i>Reduced Pareto selection strategy.</i> Selects only two candidates, the first and the last point on the Pareto front.	

without improvement, this indicates a possible discovery of the global minimum, requiring further refinement.

**Multi-level candidate selection using different  $\varepsilon$  values.** In [24, 26], two alternative multi-level techniques are proposed for the candidate selection procedure, involving three different levels:

- Level 2: The DIRECT-type algorithm is executed with the usual settings, employing  $\varepsilon = 10^{-5}$ .
- Level 1: The selection is limited to 90% of  $\bar{D}_i^k \in \mathcal{P}^k$ , excluding 10% of the candidates with the largest measure. In this level,  $\varepsilon = 10^{-7}$  is used.
- Level 0: The selection is limited to 10% of the candidates with the largest measure, disregarding those excluded at level 1. Here,  $\varepsilon = 0$  is used.

Both strategies are recommended in the study cycle through these levels using a combination of the “W-cycle”: 21011012. One of the methods [26] employs a fixed  $\varepsilon = 10^{-4}$  value at all levels, while the other [24] adheres to the rules mentioned above.

**Globally-biased selection.** In the works [34, 35], a two-phase approach with global bias was introduced. The algorithm effectively determines the adequacy of exploring a local optimum by employing the globally biased scheme. It terminates the local phase (referred to as the “usual” phase) to prevent wasteful function evaluations by excessive local refinement. Upon stopping the usual phase, the algorithm seamlessly transitions into a global phase, wherein the hyper-rectangles chosen for further exploration must meet a minimum size requirement. This globally biased phase continues until a better minimum point is discovered or a maximum number of “global iterations” is reached. Subsequently, the algorithm reverts back to the usual phase. The search process alternates between these two phases, namely, the usual phase and the globally-biased phase, until a specified stopping condition is fulfilled.

**Two-phase (Global-Local) selection.** In the work [46], a two-phase selection approach has been introduced. This approach expands the set of previously obtained POCs by incorporating additional candidates based on their proximity to the current best minimum point  $\mathbf{x}^{\min}$ . This expansion is performed by conducting a selection process using calculated distances (instead of aggregated function values) between the current best minimum point and all other candidates:

$$\mathcal{F}_k^i = \|\mathbf{x}_m^i - \mathbf{x}^{\min}\|_2. \quad (14)$$

By including candidates that are closer to the current minimum point, this step facilitates faster and more extensive exploration around the current minimum point.

## 2.4 Acceleration through hybridization techniques

To our knowledge, three hybridization strategies have been proposed for DIRECT-type algorithms [14, 16, 27, 35].

The first strategy, originally suggested by the author of the original DIRECT [16], was later refined and improved in a work [35]. The concept behind this strategy involves performing a local search only when the algorithm achieves an improvement in the best current solution value, denoted  $f^{\min}$ . The best current solution  $f^{\min}$  can be updated using a local search method or a more suitable direct-type algorithm that enables faster local refinement.

The second strategy [14] operates similarly to the first one. However, instead of performing a local search from a single starting point, this strategy employs a clustering algorithm to identify multiple appropriate starting points. The following steps are executed within this suggested method:

- The DIRECT-type algorithm is run for a fixed number of function evaluations, typically set at  $100n + 1$  as the default.
- The sampled points are analyzed using an adaptive clustering algorithm to determine the optimal number of clusters. Subsequently, a local search is performed from the best point within each cluster.
- Additionally, the DIRECT-type algorithm is run again.
- If the DIRECT-type algorithm improves  $f^{\min}$ , a final local search is performed from the best point.

In the third, aggressive strategy [27], initiate a local search from the midpoint of each POC. However, this approach has faced significant criticism for potentially generating excessive local searches, as many starting points may converge to the same local optimum.

### 3 GENDIRECT optimization software

This section describes the generalized algorithmic framework **DIRECT**. Fig. 1, illustrates the main architecture of the developed **GENDIRECT**. Specifically, there are three large boxes in Fig. 1, which represent the construction of the main **DIRECT**-type algorithmic steps within **GENDIRECT**:

1. The construction of partitioning and sampling scheme.
2. The construction of the selection scheme.
3. The construction of a hybridization scheme.

The following subsections will provide a detailed exploration of how to effectively utilize **GENDIRECT** using the **MATLAB** command line interface and the dedicated graphical user interface (GUI).

#### 3.1 Utilizing **GENDIRECT** through the command line interface.

With **GENDIRECT**, users can swiftly and effectively establish and solve global optimization problems by constructing a **DIRECT**-type algorithm via the **MATLAB** command line interface. All relevant problem information is consolidated into a unified **MATLAB** structure, which is then passed to the solver to extract the required data.

For the **GENDIRECT** format, the solution process begins by generating the following structure:

```
alg = GENDIRECT();
```

The algorithm takes in a structured input that includes the optimization problem, dimension, lower and upper bounds, and a target value (if applicable). Here is an example code snippet illustrating how these parameters can be set:

```
alg.Problem.f      = 'objfun';    % Objective function
alg.Problem.n      = n;          % Dimension
alg.Problem.x_L    = zeros(n, 1); % Lower bounds
alg.Problem.x_U    = ones(n, 1); % Upper bounds
alg.Problem.fgoal  = 0.01;       % Optimal value set as target
alg.Problem.info   = false;      % Extract info from problem
```

If the `alg.Problem.info` parameter is set to 'true', the algorithm retrieves all the relevant information about the objective function from the 'objfun' problem.

As we utilize test problems provided by **DIRECTGOLib v2.0** [49], the stored information encompasses both the problem structure and the objective function. Consequently, the algorithms automatically extract all essential details from the given problem, including:

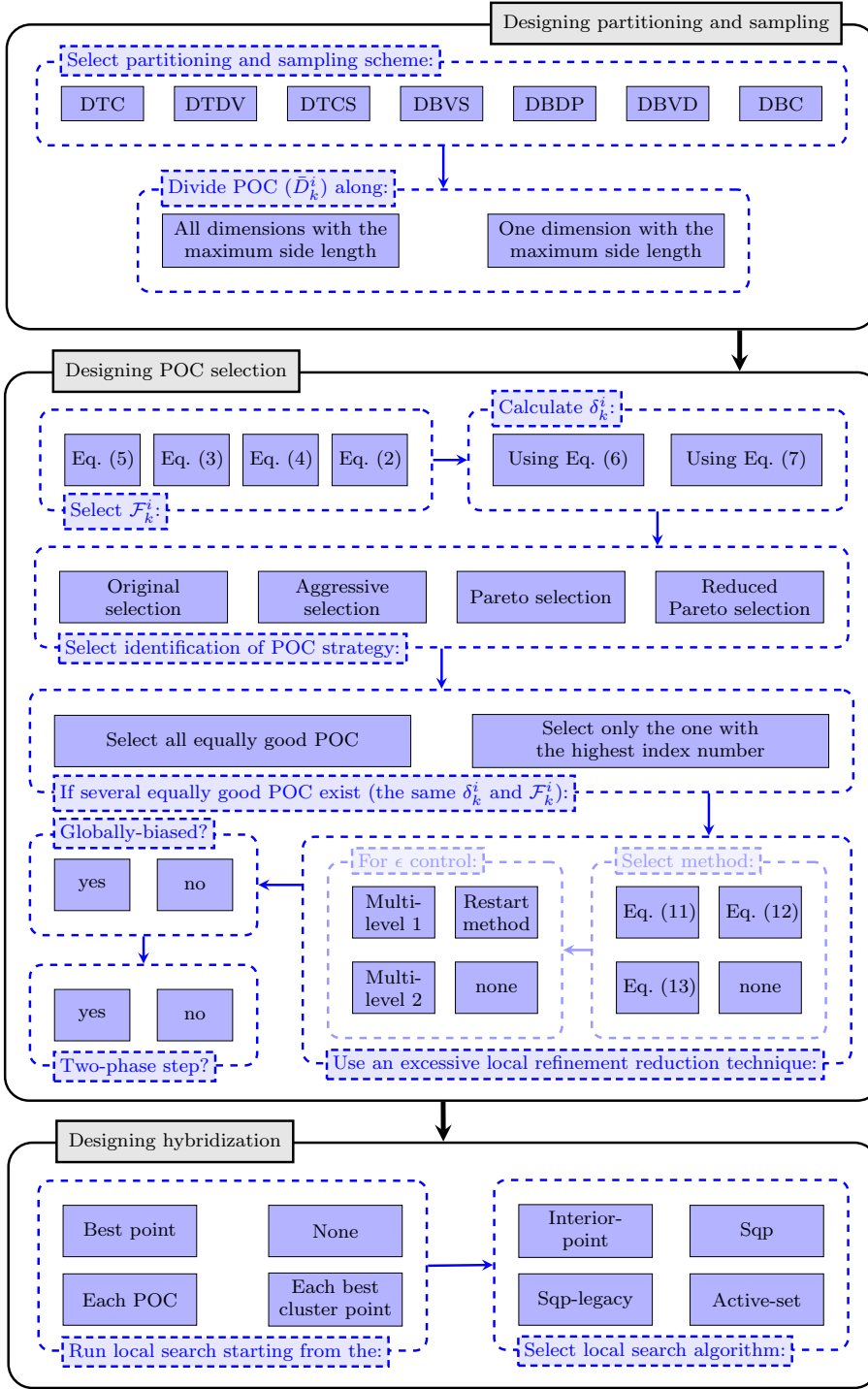


Fig. 1 A flowchart for constructing DIRECT-type algorithm in GENDIRECT.

- The dimensionality of the problem;
- The lower and upper bounds for each variable;
- The objective function value of the known solution;
- The solution point.

For further guidance on the utilization of `DIRECTGOLib v2.0`, additional information can be found in references [44, 47].

Users who want to customize the default algorithmic settings should utilize the `optParam` structure:

```
alg.optParam.maxevals = 100; % Maximal number of evaluations
alg.optParam.maxits   = 100; % Maximal number of iterations
alg.optParam.showits  = true; % Show iteration status
```

The next step involves constructing the algorithm using the procedures described in Table 3.

After completing these steps, the algorithm is ready to solve the given problem using the following line of code:

```
Results = alg.solve;
```

Once the algorithm completes its computations, it returns the `Results` structure, which contains the optimization outcomes.

The subsequent subsections will outline the process of constructing `DIRECT`-type algorithmic steps.

### 3.1.1 Designing partitioning and sampling scheme

To create a combination of `DIRECT`-type algorithms, the user needs to integrate components that determine the division and sampling strategy of the optimization domain. The core framework for constructing the partitioning and sampling strategy is illustrated in the top block of Fig. 1. The subsequent command lines illustrate how to configure the partitioning strategy of the original `DIRECT` algorithm:

```
alg.Partitioning.Strategy = 'DTC';
alg.Partitioning.SubSides = 'All';
```

As a result of the given partitioning and sampling scheme in Fig. 1, there are 14 possible combinations in `GENDIRECT`.

### 3.1.2 Designing the selection scheme

Once the partitioning and sampling strategy has been established, the subsequent task is to determine the POC selection scheme. Here is an example that illustrates the parameter values required for performing POC selection introduced in the original `DIRECT` algorithm:

**Table 3** The parameters of GENDIRECT used to construct DIRECT-type algorithms, with default values highlighted in blue.

Step	Parameter	Description
Partitioning	Strategy	Specify partitioning and sampling scheme (see Table 1): <b>DTC</b> , <b>DTDV</b> , <b>DTCS</b> , <b>DBVS</b> , <b>DBDP</b> , <b>DBVD</b> , or <b>DBC</b> .
	SubSides	Specify subdivision strategy for multiple longest sides (see Section 2.2): <b>One</b> or <b>ALL</b> .
Selection	AggrFuncVal	Specify strategy for a aggregated function value: <b>Midpoint</b> (Eq. (2)), <b>Minimum</b> (Eq. (3)), <b>Mean</b> (Eq. (4)) or <b>MidMin</b> (Eq. (5)).
	CandMeasure	Specify strategy for a measure: <b>Diagonal</b> (Eq. (6)), or <b>LongSide</b> (Eq. (7)).
	Strategy	Specify selection scheme (see Table 2): <b>Original</b> , <b>Aggressive</b> , <b>Pareto</b> , or <b>RedPareto</b> .
	EqualCand	Specify behavior for equally good POC: <b>All</b> or <b>One</b> .
	SolRefin	Specify excessive local refinement reduction technique: <b>Min</b> (Eq. (11)), <b>Median</b> (Eq. (12)), <b>Average</b> (Eq. (13)) or <b>Off</b> .
	Ep	Specify the value for $\varepsilon$ (Eqs. (11), (12), (13)): <b><math>10^{-4}</math></b> .
	ControlEp	Specify control technique for $\varepsilon$ (see Section 2.3.4): <b>Off</b> , <b>Restart</b> , <b>MultiLevel1</b> or <b>MultiLevel2</b> .
	GloballyBiased	Enable globally-biased POC selection (see Section 2.3.4): <b>Off</b> or <b>On</b> .
	TwoPhase	Enable two-phase selection of POC using Distances (Eq. (14))(see Section 2.3.4): <b>Off</b> or <b>On</b> .
	Strategy	Specify hybridization strategy (see Section 2.4): <b>Off</b> , <b>Single</b> , <b>Clustering</b> or <b>Aggressive</b> .
Hybridization	LocalSearch	Specify derivative-free local search subroutine: <b>interior-point</b> , <b>sqp</b> , <b>sqp-legacy</b> or <b>active-set</b> .
	MaxIterations	Specify the maximum iteration limit for a single local search subroutine call: <b>1000</b> .
	MaxEvaluations	Specify the maximum function evaluation limit for a single local search subroutine call: <b>3000</b> .

```

alg.Selection.AggrFuncVal = 'Midpoint';
alg.Selection.CandMeasure = 'Diagonal';
alg.Selection.Strategy = 'Original';
alg.Selection.EqualCand = 'All';
alg.Selection.SolRefin = 'Min';
alg.Selection.Ep = 0.0001;
alg.Selection.ControlEp = 'Off';
alg.Selection.GloballyBased = 'Off';
alg.Selection.TwoPhase = 'Off';

```

When the two-phase selection step is enabled, as demonstrated in the following code snippet:

```
alg.Selection.TwoPhase = 'On';
```

the algorithm uses the designed selection scheme ('alg.Selection') to expand the set of promising candidate solutions (POC) based on the calculated distances obtained using Eq. (14). It is easy to calculate in Fig. 1, there are 4096 different combinations for the selection steps of POC in **GENDIRECT**.

### 3.1.3 Designing hybridization scheme

In the third block of Fig. 1, users are required to select the desired hybridization technique. There are only 13 possible combinations available in this block.

For example, to specify a hybridization scheme that utilizes a strategy calling an SQP local search (parameter **sqp**) subroutine only when an improvement in the best current solution is achieved (parameter **Single**), the following code can be used:

```
alg.Hybridization.Strategy = 'Single';
alg.Hybridization.LocalSearch = 'sqp';
```

## 3.2 Utilizing **GENDIRECT** through the graphical user interface

**GENDIRECT** is also accessible through the graphical user interface (GUI) of **DIRECTGO**. This GUI enables users to use **GENDIRECT** without requiring prior programming or algorithmic knowledge. To access the **GENDIRECT** tool, users can navigate to the **MATLAB APPS** menu on the toolbar. Within **DIRECTGO**, the generalized **DIRECT** algorithm (**GENDIRECT**) can be selected from the algorithm drop-down menu.

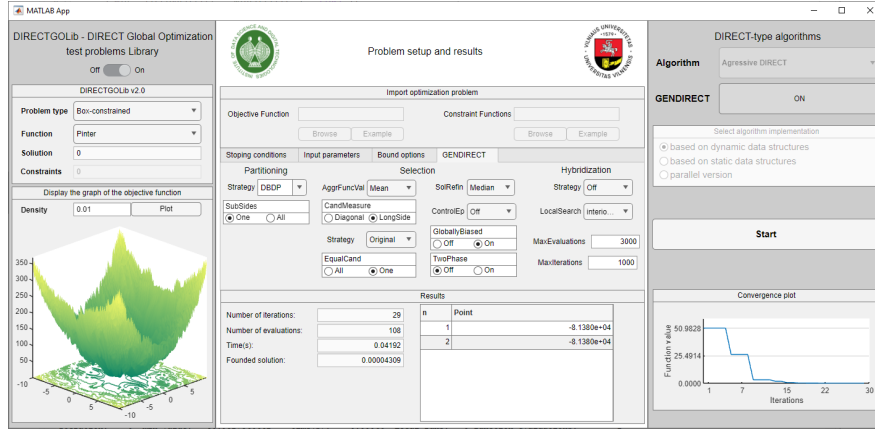
The graphical interface of the main toolbox window **DIRECTGO** is depicted in Fig. 2. The **GENDIRECT** window is centrally located in the GUI and facilitates the construction of the **DIRECT** algorithm by providing user-friendly functionalities. For more comprehensive details of **DIRECTGO**, see [47].

## 3.3 Remarks regarding the extension of **GENDIRECT**

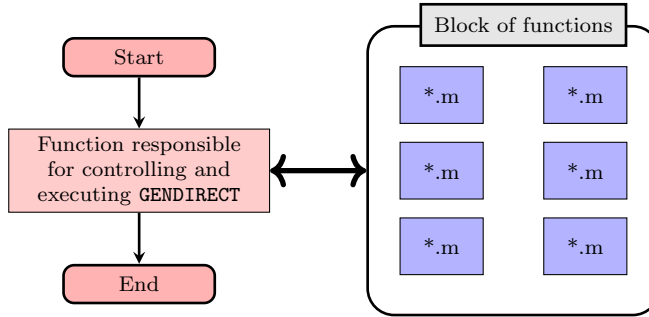
**GENDIRECT** comprises two primary components, as illustrated in Fig. 3. Firstly, a function block encompasses various implementations of the steps involved in **DIRECT**-type algorithms. Secondly, the control structure ensures the seamless connection of algorithm components, facilitating the execution of the algorithm.

If a researcher intends to integrate a newly proposed step into **GENDIRECT**, the function should be added to the function block. Ensuring that the implemented function adheres to the existing code's style is important. Subsequently, in the control function of **GENDIRECT**, the newly created function should be incorporated accordingly, allowing **GENDIRECT** to utilize it effectively.





**Fig. 2** A snapshot of the graphical user interface (GUI) of GENDIRECT in the DIRECTGO software package.



**Fig. 3** The framework of the generalized DIRECT algorithm system (GENDIRECT)

## 4 Simulation results and in-depth analysis

This section presents an analysis of the experimental results for newly developed improved algorithms and their performance evaluation using GENDIRECT.

### 4.1 An overview of benchmark test problems

We employed a comprehensive set of 324 benchmark test functions to thoroughly evaluate the newly proposed GENDIRECT algorithm. These test problems were sourced from the latest version of the DIRECTGOLib v2.0 library [43], which is built within the MATLAB environment. The DIRECTGOLib v2.0 integrates ten libraries and collections of well-established and recently developed test problems.

In Table 4, we present a summary of DIRECTGOLib v2.0 and its constituent libraries. The table provides essential details, including references, publication years, the pool of problems, and the counts of scalable, separable, and multi-modal problems. Specifically, the table comprises 136 test problems with fixed dimensions and 188 test benchmarks that can be adjusted to any dimension

size ( $n$ ). For these test problems, we consider instances with variables set at  $n = 2, 5$ , and  $10$ . However, it is worth noting that some functions, such as certain CEC functions [23, 55], are not applicable in all dimensions.

In our study, we thoroughly examined a total of 634 test problems available in **DIRECTGOLib v2.0** to ensure comprehensive and robust evaluations of the proposed algorithmic framework **GENDIRECT**.

**Table 4** Compilation of test problems from various libraries in the latest version of the **DIRECTGOLib v2.0** for box-constrained global optimization.

Source	Year	Problems			
		Total	Scalable	Seperable	Multi-modal
Hedar, [12]	2005	31	17	8	23
Hansen et al., [11]	2009	24	24	5	14
Jamil et al., [15]	2013	167	69	49	127
Gavana, [9]	2013	193	76	64	156
Surjanovic et al., [53]	2013	50	23	11	40
Liang et al., [23]	2014	27	27	5	24
Wu et al., [55]	2017	20	20	0	19
Oldenhuis, [32]	2020	41	12	5	33
Layeb, [1]	2022	18	18	2	16
Kudela et al., [20]	2022	8	8	8	8
Stripinis et al., [43]	2023	324	188	97	261

In order to ensure that the global minimum point does not coincide with the initial sampling point in any tested algorithm, we employ shift operations. In other words, we randomly shift the solutions in the  $X$ -space. This involves transforming a given point  $\mathbf{x}$  into  $\hat{\mathbf{x}}$  using the following equation:

$$\hat{x}_j = \min \{ \max \{ x_j - \rho_j \lambda \vec{x}_j, a_j \}, b_j \}, \quad j = 1, \dots, n. \quad (15)$$

Here,  $\vec{\mathbf{x}}$  is a randomly distributed random direction vector generated using the Mersenne-Twister pseudorandom generator, and  $\lambda$  is a step size that serves two important purposes:

- It prevents the global optima from moving outside of the feasible region.
- It allows for a more efficient placement of the solution within the problem domain, considering that different problems may have significantly different domain sizes.

The value of  $\lambda$  is calculated by solving the following linear programming problem:

$$\begin{aligned} \max \quad & \lambda \\ \text{s.t.} \quad & \mathbf{x}^* + \lambda \vec{\mathbf{x}} \geq \mathbf{a} \\ & \mathbf{x}^* + \lambda \vec{\mathbf{x}} \leq \mathbf{b} \end{aligned} \quad (16)$$

The shift operation introduces the possibility of regions outside the original feasible range  $[\mathbf{a}, \mathbf{b}]$  where, in certain instances, points with lower function values than the global optimum within the original feasible range may exist. To tackle this issue, the transformed vector  $\hat{\mathbf{x}}$  (15) is restricted to lie within the range  $[\mathbf{a}, \mathbf{b}]$  using min-max functions.

Nevertheless, one drawback of this approach is that the functions become “flat” in areas where the min-max restriction is applied. These flat regions increase in size as the value of  $\lambda$  increases. To address this concern, we opted to limit the range of the randomly generated shift vector by assigning a uniformly distributed random multiplication rate  $\rho_j \in [0, 0.1]$  to each dimension  $j = 1, \dots, n$ .

For convenient access to all test problems utilized in this paper and to replicate the random shift vectors, we created a dedicated **MATLAB** script in the “Scripts/MPC” directory of the GitHub repository (<https://github.com/blockchain-group/DIRECTGO>). These scripts serve as valuable tools for reproducing the findings presented in this investigation and for comparing and evaluating newly developed algorithms.

#### 4.2 Setup and fundamental basis for algorithm comparison

All computations were executed on an Intel(R) Core<sup>TM</sup> i5-10400 @ 2.90GHz Processor running **MATLAB** R2023a. The algorithms’ solutions were compared with the globally optimal solution for each problem, and we considered the solver successful when the objective function value of a solution was within 0.01% of the global optimum.

For all analytical test cases with a known global optimum  $f^*$ , we employed a stopping criterion based on the percent error ( $pe$ ), as defined below:

$$pe = 100 \times \begin{cases} \frac{f(\mathbf{x}) - f^*}{|f^*|}, & f^* \neq 0 \\ f(\mathbf{x}), & f^* = 0 \end{cases} \quad (17)$$

The algorithms were terminated under the following conditions:

- When the ( $pe$ ) became smaller than  $\varepsilon_{pe} = 0.01$ .
- When the number of function evaluations exceeded the prescribed limit  $M_{\max} = n \times 10^5$ .
- When the execution time exceeded  $T_{\max} = 30$  CPU minutes. In such cases, the final result was set to  $n \times 10^5$  to facilitate further processing of the result.

#### 4.3 Algorithm design in GENDIRECT

Considering that the developed **GENDIRECT** software allows for a large number of combinations, identifying the most effective ones may require a substantial amount of time and effort. Therefore, we cannot guarantee that the algorithms presented are the most efficient within **GENDIRECT**. Furthermore, the benchmark set includes numerous distinct problems, such as discontinuous, non-differentiable, multi-modal, non-symmetric, and plateau functions. It is improbable that a single combination will be the most efficient for all of these diverse problem types.

According to the no-free lunch theorem for optimization [54], there exists no universal optimization algorithm that performs optimally on all types of optimization problems. As a result, certain modifications and additions to specific algorithms may not enhance performance on all problems and could even lead to a decline in performance in certain cases. Therefore, the most optimal approach would involve leveraging machine learning-enhanced automated

algorithm selection techniques [19] to generate algorithms tailored to specific problems. However, this avenue remains a part of our future work and has yet to be explored.

To showcase the benefits of **GENDIRECT** software, we conducted an experiment involving five existing **DIRECT**-type algorithms: **1-DTC-GL** [51], **HALRECT-IA** [45], **MrDIRECT** [26], **BIRMIN** [35], and **DIRMIN** [28]. Our aim was to improve their average performance across a designated set of test problems by introducing new algorithmic steps or substituting existing ones.

In Table 5, we present five variants for each of the five selected algorithms, with their improved versions. For pure algorithms of **DIRECT**-type, which are characterized by slow solution refinement, enhancing their performance was achieved by incorporating local search techniques. On the other hand, for hybrid methods, we made different adjustments to improve their performance. Specifically, for the **BIRMIN** algorithm, our goal was to increase the number of evaluations per iteration through enhancements, while for the **DIRMIN** algorithm, we pursued the opposite approach.

**Table 5** Description of used parameters for each selected algorithm and their improved versions in **GENDIRECT**. The blue color indicates the parameter that has been substituted or has been added.

Original algorithm parameters	1-DTC-GL	HALRECT-IA	MrDIRECT	BIRMIN	DIRMIN
Partitioning.Strategy	'DTC'	'DBC'	'DTC'	'DBDP'	'DTC'
Partitioning.SubSides	'One'	'All'	'All'	'One'	'All'
Selection.AggrFuncVal	'Midpoint'	'MidMin'	'Midpoint'	'Min'	'Midpoint'
Selection.CandMeasure	'Diagonal'	'Diagonal'	'Diagonal'	'Diagonal'	'Diagonal'
Selection.Strategy	'Pareto'	'Aggressive'	'Original'	'Original'	'Original'
Selection.EqualCand	'One'	'One'	'All'	'One'	'All'
Selection.SolRefin	'Off'	'Off'	'Min'	'Min'	'Min'
Selection.Ep	—	—	0.0001	0.0001	0.0001
Selection.ControlEp	'Off'	'Off'	'MultiLevel1'	'Off'	'Off'
Selection.GloballyBiased	'Off'	'Off'	'Off'	'On'	'Off'
Selection.TwoPhase	'On'	'Off'	'Off'	'Off'	'Off'
Hybridization.Strategy	'Off'	'Off'	'Off'	'Single'	'Aggressive'
Hybridization.LocalSearch	—	—	—	'interior-point'	'interior-point'
Hybridization.MaxIterations	—	—	—	1000	1000
Hybridization.MaxEvaluations	—	—	—	3000	3000
Improved algorithm parameters	1-DTC-GL	HALRECT-IA	MrDIRECT	BIRMIN	DIRMIN
Partitioning.Strategy	'DTC'	'DBC'	'DTC'	'DBDP'	'DTC'
Partitioning.SubSides	'One'	'All'	'All'	'All'	'All'
Selection.AggrFuncVal	'Midpoint'	'MidMin'	'Midpoint'	'Min'	'Midpoint'
Selection.CandMeasure	'Diagonal'	'Diagonal'	'Diagonal'	'Diagonal'	'LongSide'
Selection.Strategy	'Pareto'	'Aggressive'	'Original'	'Pareto'	'Original'
Selection.EqualCand	'One'	'One'	'All'	'One'	'One'
Selection.SolRefin	'Off'	'Off'	'Min'	'Min'	'Median'
Selection.Ep	—	—	0.0001	0.0001	0.0001
Selection.ControlEp	'Off'	'Off'	'MultiLevel1'	'Off'	'Off'
Selection.GloballyBiased	'Off'	'Off'	'Off'	'On'	'Off'
Selection.TwoPhase	'On'	'Off'	'Off'	'Off'	'Off'
Hybridization.Strategy	'Single'	'Aggressive'	'Clustering'	'Single'	'Aggressive'
Hybridization.LocalSearch	'sqp'	'sqp'	'sqp'	'sqp'	'interior-point'
Hybridization.MaxIterations	1000	1000	1000	1000	1000
Hybridization.MaxEvaluations	3000	3000	3000	3000	3000

It is essential to note that the construction of the original algorithms within **GENDIRECT** may not always produce identical results to the implementations provided in **DIRECTGO** [48]. The discrepancy in the results can be attributed to the numerical tolerances used in the implementations, which play a critical role in the outcome. For instance, authors might employ rounding on hyper-rectangle measure sizes, enabling them to group extremely small hyper-rectangles together.

Additionally, they might consider two function values identical if their difference is below a certain threshold. These variations in the implementations can significantly impact the selection of POCs.

#### 4.4 Results and discussions

In this section, we conduct a performance evaluation of ten DIRECT-type algorithms, five of which are newly generated with GENDIRECT. The experimental results presented in this evaluation can also be accessed digitally in the “Results/MPC” directory of the GitHub repository, available at <https://github.com/blockchain-group/DIRECTGO>.

##### 4.4.1 Comparison of success rates and function evaluations utilization

Table 6 provides an overview of the success rates achieved by the ten DIRECT-type approaches considered on various subsets of the DIRECTGOLib v2.0 test problems. In particular, improvements that effectively improve the performance of the original algorithm are highlighted in green, while those that lead to deteriorating results are marked in red. The most remarkable enhancements in success rates were observed in the case of the algorithm that performed worst in this study (MrDIRECT) after applying the improvements. Its enhanced version yielded a remarkable increase in the success rate of 15.78%. Moreover, the most significant improvements were evident in the resolution of uni-modal problems, where the pure MrDIRECT version failed to locate the desired solutions within the allocated evaluation budget in 32.41% fewer instances.

**Table 6** Comparison of the success rates of different algorithms in solving test problems with various characteristics.

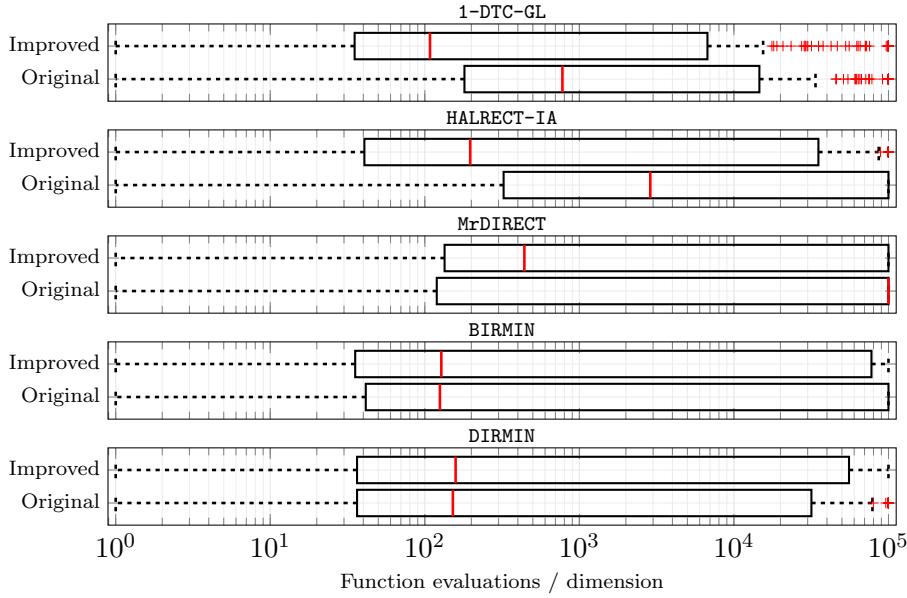
Algorithm	Percentage of solved problems						
	Overall	Separability		Multi-modality		Scalability	
		+	−	+	−	+	−
Impr. 1-DTC-GL	82.18	91.71	77.62	77.30	98.62	78.92	94.12
Orig. 1-DTC-GL	80.60	91.71	75.29	75.66	97.24	76.91	94.12
Impr. HALRECT-IA	78.08	87.32	73.66	72.19	97.93	73.69	94.12
Orig. HALRECT-IA	67.35	76.59	62.94	62.58	83.45	62.45	85.29
Impr. MrDIRECT	64.20	78.05	57.58	55.42	93.79	59.84	80.15
Orig. MrDIRECT	48.42	65.85	40.09	44.58	61.38	43.17	67.65
Impr. BIRMIN	75.08	83.90	70.86	68.30	97.93	70.68	91.17
Orig. BIRMIN	70.66	82.43	65.03	63.60	94.48	65.06	91.17
Impr. DIRMIN	76.34	84.39	72.49	70.34	96.55	71.28	94.85
Orig. DIRMIN	77.76	84.88	74.36	72.19	96.55	73.09	94.85

Among the pure DIRECT-type algorithms, the 1-DTC-GL algorithm exhibited the lowest increase in success rates. When considering the allocated budget for function evaluations, the improved algorithm 1-DTC-GL failed to provide a solution to the 113 problems, while the original version struggled with the 123 problems. An

important observation is that the original algorithm **1-DTC-GL** performed quite well, surpassing the overall performance of the improved versions of other less efficient algorithms.

The enhancements in the hybrid algorithms resulted in increased success rates only for the **BIRMIN** algorithm, whereas the success rates for the **DIRMIN** algorithms exhibited a slight deterioration in most of the subsets considered. Despite the improvement achieved in the **BIRMIN** algorithm, it still remained outperformed by both versions of the **DIRMIN** algorithm in almost all cases.

Fig. 4 presents a box plot that compares algorithms based on function evaluations per dimension on all test problems. An important distinction between pure and hybrid algorithms is that pure algorithms generally require more function evaluations, even for relatively simple optimization problems. On the contrary, hybrid algorithms demonstrate the ability to solve such problems quickly and efficiently. Among the algorithms, almost all hybrid algorithms achieved similar lowest first-quartile values, indicating that these methods could solve at least 25% of the test problems faster than pure algorithms. Specifically, four algorithms (original and improved **DIRMIN**, improved **BIRMIN**, and improved **1-DTC-GL**) were in the lowest first quartile. On the contrary, the original **HALRECT-IA** and original **1-DTC-GL** algorithms exhibited the worst first-quartile performance, each requiring approximately nine and six times more function evaluations, respectively, than the best-performing algorithm, **DIRMIN**.



**Fig. 4** Box plot graphical comparison of algorithms performance based on function evaluations per dimension across all test problems.

The improved algorithm **1-DTC-GL** demonstrated the best median value, while its pure counterpart, the original version **1-DTC-GL**, had the third worst

median value in these studies. The addition of the local search procedure to the 1-DTC-GL algorithm reduced the median value by nearly eight times, resulting in a significant improvement in its performance. Interestingly, the median value of the original algorithm **MrDIRECT** is equal to the maximum number of function evaluation budgets ( $M_{\max}$ ), indicating that the algorithm could not solve more than half of the test problems. However, its improved version exhibited a significantly higher median value. When comparing the third-quartile values, four algorithms reached the  $M_{\max}$  value in the third quartile, suggesting that these algorithms could not solve more than 25% of the problems. Only six algorithms achieved values lower than the maximum evaluation budget. Among these, the improved algorithm 1-DTC-GL achieved the lowest third-quartile value, approximately half that of the second-best pure algorithm, the original algorithm 1-DTC-GL.

#### 4.4.2 Analysis of results across different subsets of problems

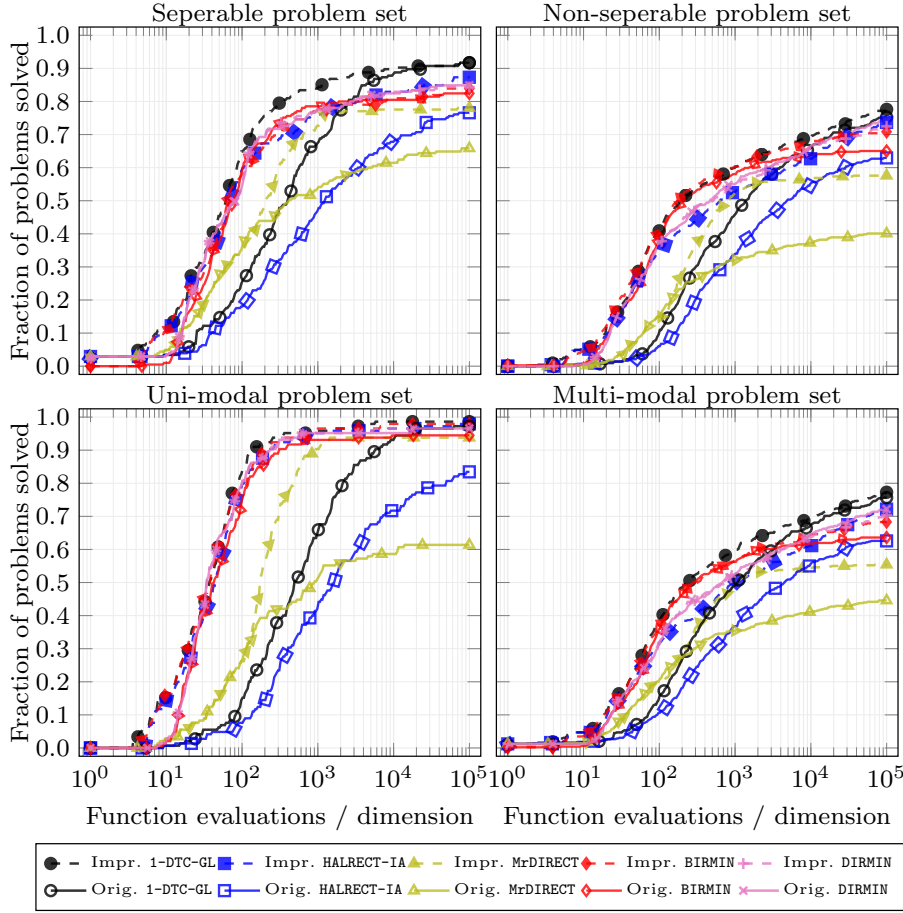
The data profiles [31] depicted in Fig. 5 showcase how all algorithms perform on test problems with various properties of **DIRECTGOLib v2.0**. These profiles provide a comprehensive view of algorithm performance across different types of problems. Meanwhile, the data profiles in Fig. 6 offer an overall ranking of the algorithms on all test problems, providing a more focused perspective on their performance in a broader context.

Hybridization of pure DIRECT-type algorithms significantly impacts the results, particularly when dealing with straightforward uni-modal or separable test problems. The inclusion of a local search procedure proves to be particularly advantageous for uni-modal problems, as it accelerates the convergence speed to reach optimal solutions more efficiently. On the other hand, pure DIRECT-type algorithms might prioritize the global search and exhaust the evaluation budget without locating the solution within the prescribed accuracy. As a result, the curves of the improved versions of 1-DTC-GL, HALRECT-IA, and **MrDIRECT** demonstrate significantly better performance than the original versions, especially for small evaluation budgets ( $\leq 1000 \times n$ ). However, it is worth noting that the most successful pure DIRECT-type algorithm, 1-DTC-GL, eventually achieves nearly identical performance within the maximum evaluation budget, regardless of whether the problems are separable or uni-modal.

The improved hybrid algorithm **BIRMIN** exhibits slightly lower performance within a small evaluation budget ( $M_{\max} \leq n \times 10^2$ ). However, as the evaluation budget increases ( $M_{\max} \geq n \times 10^4$ ), the improved version outperforms the original version. This difference in performance becomes particularly evident when the algorithm is applied to non-separable or multi-modal test problems.

On the other hand, the curves of the two versions of the hybrid algorithm **DIRMIN** are almost indistinguishable within a smaller evaluation budget ( $M_{\max} \leq 2n \times 10^4$ ). However, within a larger evaluation budget, the original algorithm **DIRMIN** exhibits slightly better performance.

Based on the four graphs in Fig. 5 and the overall ranking of the algorithms in Fig. 6, a consistent conclusion can be drawn: the performance of the improved and original 1-DTC-GL algorithm is the most efficient or at least comparable to the best-performing algorithm. Analyzing the curves, it is evident that the improved algorithm 1-DTC-GL exhibits the highest efficiency rates across all



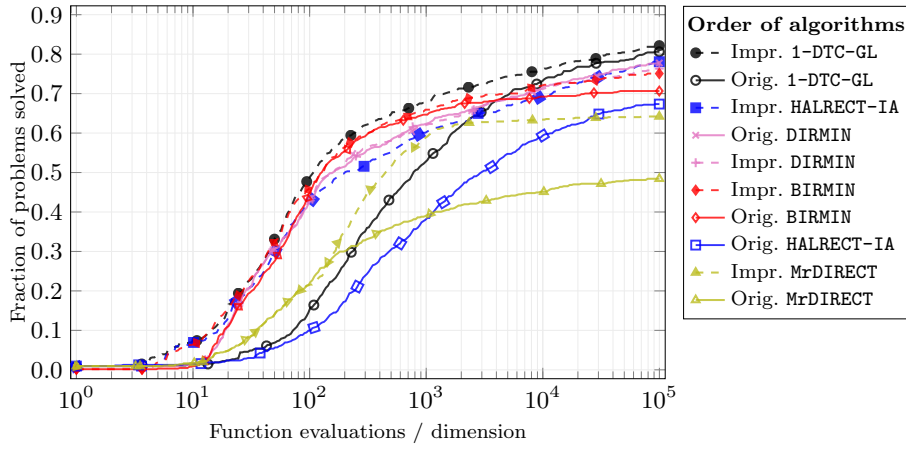
**Fig. 5** Data profiles: The horizontal axis represents the number of function evaluations per dimension, while the vertical axis represents the fraction of solved problems

graphs compared to other algorithms within any evaluation budget in  $[0, n \times 10^5]$ . Although the original performance of 1-DTC-GL becomes competitive, it requires a significant budget for the evaluations of functions ( $M_{\max} \geq n \times 10^4$ ). Overall, the improved performance of the 1-DTC-GL algorithm remains competitive, requiring fewer function evaluations to achieve the desired optimal value in most test functions.

#### 4.4.3 Statistical analysis of the results

To validate the results and comparisons between algorithms, as well as to evaluate the significance of improvements achieved by GENDIRECT, we conducted the Friedman mean rank test [7] and the non-parametric Wilcoxon signed test [13] at a significance level of 5%. A  $p$ -value greater than 0.05 indicates that the difference in results between methods is statistically insignificant.





**Fig. 6** Data profiles: The horizontal axis represents the number of function evaluations per dimension, while the vertical axis represents the fraction of solved problems

Table 7 displays the Friedman mean rank values for considered algorithms, utilizing the founded solution values within different evaluation budgets for all test problems. The results reveal that the improved versions consistently outperform their original counterparts in all budgets, except for the algorithm DIRMIN, where the original version obtained a higher rank in one specific evaluation budget ( $M_{\max}=n \times 10^5$ ). The improvements made to the algorithms have resulted in performance gains ranging from small to significant, as indicated by the higher mean rank values.

**Table 7** Friedmann mean rank values with different objective function evaluation budgets.

Algorithm	Function evaluation budget ( $M_{\max}$ )			
	$n \times 10^2$	$n \times 10^3$	$n \times 10^4$	$n \times 10^5$
Impr. 1-DTC-GL	4.7610	4.6128	4.6128	4.6601
Orig. 1-DTC-GL	6.0095	5.3375	5.3375	4.7831
Impr. HALRECT-IA	5.5670	5.5229	5.5229	5.3249
Orig. HALRECT-IA	7.2752	7.1372	7.1372	6.1672
Impr. MrDIRECT	5.3730	5.2500	5.2500	5.8028
Orig. MrDIRECT	5.3730	6.8099	6.8099	7.1435
Impr. BIRMIN	4.6151	4.6562	4.6562	5.1333
Orig. BIRMIN	5.0103	5.0765	5.0765	5.6356
Impr. DIRMIN	5.4219	5.2886	5.2886	5.2492
Orig. DIRMIN	5.5938	5.3084	5.3084	5.1002

Table 8 presents the  $p$ -values obtained by comparing the improved algorithms with their original counterparts, using the solutions found within four evaluation budgets on all test problems. For the improved algorithm 1-DTC-GL, there is strong statistical evidence that the improved version significantly outperforms

the original version within a small evaluation budget ( $M_{\max} \leq n \times 10^3$ ). However, as the evaluation budget increases ( $M_{\max} > n \times 10^3$ ), the higher  $p$ -values suggest that the significance of the improvement decreases and the difference between the improved and original versions becomes less statistically significant. In contrast, the situation is different for the other two pure DIRECT-type algorithms. The improved versions of HALRECT-IA and MrDIRECT show no significant improvement compared to the original versions at  $M_{\max} = n \times 10^2$ . However, for larger evaluation budgets, the  $p$ -values are low, indicating that the improvements are statistically significant.

For the BIRMIN algorithm, the  $p$ -values are low, indicating that the improvement of the improved version of BIRMIN compared to the original version is statistically significant in all these budgets. Regarding DIRMIN algorithm, we can conclude that the improved version of the algorithm is statistically better if the evaluation budgets are  $M_{\max} = n \times 10^2$  and  $M_{\max} = n \times 10^4$ .

**Table 8** Wilcoxon signed test  $p$ -values at 5% significance level, comparing improved vs. original algorithms across various objective function evaluation budgets.

Algorithm	Function evaluation budget ( $M_{\max}$ )			
	$n \times 10^2$	$n \times 10^3$	$n \times 10^4$	$n \times 10^5$
1-DTC-GL	$6.3325 \times 10^{-3}$	$2.4241 \times 10^{-8}$	$2.6994 \times 10^{-1}$	$5.8529 \times 10^{-1}$
HALRECT-IA	$2.3254 \times 10^{-1}$	$4.0565 \times 10^{-13}$	$5.3640 \times 10^{-10}$	$3.3265 \times 10^{-13}$
MrDIRECT	$1.0000 \times 10^0$	$2.0141 \times 10^{-34}$	$4.5374 \times 10^{-40}$	$5.9712 \times 10^{-34}$
BIRMIN	$2.5863 \times 10^{-2}$	$2.8668 \times 10^{-9}$	$1.0548 \times 10^{-12}$	$1.5552 \times 10^{-15}$
DIRMIN	$3.0296 \times 10^{-8}$	$4.6683 \times 10^{-1}$	$8.2874 \times 10^{-3}$	$4.4600 \times 10^{-4}$

## 5 Conclusions and future works

This study introduces a novel generalized DIRECT-type algorithmic framework known as GENDIRECT, for derivative-free global optimization. The proposed framework empowers users to construct a wide range of DIRECT-type algorithms. Such innovative work can foster the development of new DIRECT-type algorithms and help identify the most suitable algorithm for various practical applications.

To demonstrate the efficiency of GENDIRECT, we enhanced five selected DIRECT-type algorithms with the goal of improving their performance and solving global optimization problems more effectively. Evaluation of these constructed algorithms was carried out using benchmark test functions from DIRECTGOLib v2.0. The results were analyzed both graphically and statistically to gain insight into the algorithms' performance. The findings concluded that the newly developed versions of the DIRECT-type algorithms significantly outperformed their original counterparts in most cases.

In conclusion, this paper has focused on box-constrained global optimization problems, but the generalized DIRECT-type algorithmic framework (GENDIRECT) could potentially be extended to handle constrained cases as well. Furthermore, due to the numerous combinations of algorithms within GENDIRECT, manually testing all of them becomes impractical. Therefore, future research should

explore the automation of these processes using advanced machine-learning techniques. By automating the algorithmic components process, optimization can become more efficient and effective.

## Data statement

**DIRECTGOLib - DIRECT Global Optimization test problems Library** is designed as a continuously-growing open-source GitHub repository to which anyone can easily contribute. The exact data underlying this article from DIRECTGOLib v2.0 can be accessed on GitHub:

– <https://github.com/blockchain-group/DIRECTGOLib>,

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

## References

1. Abdesslem, L.: New hard benchmark functions for global optimization (2022). URL <https://www.mathworks.com/matlabcentral>. MATLAB Central File Exchange. Retrieved February 18, 2022.
2. Agrawal, P., Abutarboush, H.F., Ganesh, T., Mohamed, A.W.: Metaheuristic algorithms on feature selection: A survey of one decade of research (2009-2019). *IEEE Access* **9**, 26766–26791 (2021)
3. Baker, C.A., Watson, L.T., Grossman, B., Mason, W.H., Haftka, R.T.: Parallel global aircraft configuration design space exploration. In: A. Tentner (ed.) *High Performance Computing Symposium 2000*, pp. 54–66. Soc. for Computer Simulation Internat (2000)
4. Chiter, L.: Experimental data for the preprint "diagonal partitioning strategy using bisection of rectangles and a novel sampling scheme" (2023). DOI 10.17632/x9fpc9w7wh.2. URL <https://data.mendeley.com/datasets/x9fpc9w7wh>
5. Finkel, D.E., Kelley, C.T.: An adaptive restart implementation of direct. Technical report CRSC-TR04-30, Center for Research in Scientific Computation, North Carolina State University, Raleigh (2004)
6. 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
7. Friedman, M.: The use of ranks to avoid the assumption of normality implicit in the analysis of variance. *Journal of the American Statistical Association* **32**(200), 675–701 (1937). DOI 10.1080/01621459.1937.10503522. URL <https://www.tandfonline.com/doi/abs/10.1080/01621459.1937.10503522>
8. 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
9. Gavana, A.: Global optimization benchmarks and ampgo. [http://infinity77.net/global\\_optimization/index.html](http://infinity77.net/global_optimization/index.html). Online; accessed: 2021-07-22
10. Golovin, D., Solnik, B., Moitra, S., Kochanski, G., Karro, J., Sculley, D.: Google vizier: A service for black-box optimization. In: *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, KDD '17*, p. 1487–1495. Association for Computing Machinery, New York, NY, USA (2017). DOI 10.1145/3097983.3098043. URL <https://doi.org/10.1145/3097983.3098043>
11. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-Parameter Black-Box Optimization Benchmarking 2009: Noiseless Functions Definitions. Research Report RR-6829, INRIA (2009). URL <https://inria.hal.science/inria-00362633>
12. Hedar, A.: Test functions for unconstrained global optimization. [http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar\\_files/TestG0.htm](http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestG0.htm) (2005). Online; accessed: 2017-03-22
13. Hollander, M., Wolfe, D.: *Nonparametric Statistical Methods, Solutions Manual*. Wiley Series in Probability and Statistics. Wiley (1999). URL <https://books.google.lt/books?id=pTQFAAAACAAJ>

14. Holmström, K., Edvall, M.M.: The TOMLAB Optimization Environment, pp. 369–376. Springer US, Boston, MA (2004). DOI 10.1007/978-1-4613-0215-5\_19. URL [https://doi.org/10.1007/978-1-4613-0215-5\\_19](https://doi.org/10.1007/978-1-4613-0215-5_19)
15. Jamil, M., Yang, X.S.: A literature survey of benchmark functions for global optimisation problems. *International Journal of Mathematical Modelling and Numerical Optimisation* **4**(2), 150–194 (2013). DOI 10.1504/IJMMNO.2013.055204. URL <https://www.inderscienceonline.com/doi/abs/10.1504/IJMMNO.2013.055204>. PMID: 55204
16. 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, Dordrecht (2001)
17. 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
18. Jones, D.R., Schonlau, M., Welch, W.J.: Efficient Global Optimization of Expensive Black-Box Functions. *Journal of Global Optimization* **13**(4), 455–492 (1998). DOI 10.1023/A:1008306431147. URL <https://doi.org/10.1023/A:1008306431147>
19. Kerschke, P., Hoos, H.H., Neumann, F., Trautmann, H.: Automated Algorithm Selection: Survey and Perspectives. *Evolutionary Computation* **27**(1), 3–45 (2019). DOI 10.1162/evco\_a.00242. URL [https://doi.org/10.1162/evco\\_a.00242](https://doi.org/10.1162/evco_a.00242)
20. Kudela, J., Matousek, R.: New benchmark functions for single-objective optimization based on a zigzag pattern. *IEEE Access* **10**, 8262–8278 (2022). DOI 10.1109/ACCESS.2022.3144067
21. Kudela, J., Matousek, R.: Recent advances and applications of surrogate models for finite element method computations: a review. *Soft Computing* **26**, 13709–13733 (2022). DOI 10.1007/s00500-022-07362-8. URL <https://doi.org/10.1007/s00500-022-07362-8>
22. Larson, J., Menickelly, M., Wild, S.M.: Derivative-free optimization methods. *Acta Numerica* **28**(2010) (2019). DOI 10.1017/S0962492919000060
23. Liang, J.J., Qu, B.Y., Suganthan, P.N.: Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization. Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore **635**(2) (2013)
24. Liu, H., Xu, S., Wang, X., Wu, X., Song, Y.: A global optimization algorithm for simulation-based problems via the extended direct scheme. *Engineering Optimization* **47**(11), 1441–1458 (2015). DOI 10.1080/0305215X.2014.971777
25. Liu, Q.: Linear scaling and the direct algorithm. *Journal of Global Optimization* **56**, 1233–1245 (2013). DOI 10.1007/s10898-012-9952-x
26. Liu, Q., Zeng, J., Yang, G.: MrDIRECT: a multilevel robust DIRECT algorithm for global optimization problems. *Journal of Global Optimization* **62**(2), 205–227 (2015). DOI 10.1007/s10898-014-0241-8
27. Liuzzi, G., Lucidi, S., Piccialli, V.: A DIRECT-based approach exploiting local minimizations for the solution of large-scale global optimization problems. *Computational Optimization and Applications* **45**, 353–375 (2010). DOI 10.1007/s10589-008-9217-2
28. 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 10.1007/s10589-015-9741-9
29. Mockus, J.: On the pareto optimality in the context of lipschitzian optimization. *Informatica* **22**(4), 521–536 (2011). DOI 10.15388/Informatica.2011.340
30. Mockus, J., Paulavičius, R.: On the reduced-set pareto-lipschitzian optimization. *Computational Science and Techniques* **1**(2), 184–192 (2013)
31. Moré, J.J., Wild, S.M.: Benchmarking derivative-free optimization algorithms. *SIAM Journal on Optimization* **20**(1), 172–191 (2009). DOI 10.1137/080724083
32. Oldenhuis, R.: Test functions for global optimization algorithms. <https://github.com/rodyo/FEX-testfunctions/releases/tag/v1.5>. Online; accessed: 2023-02-22
33. 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
34. 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
35. 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

36. 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)
37. Paulavičius, R., Žilinskas, J.: Simplicial Lipschitz optimization without the Lipschitz constant. *Journal of Global Optimization* **59**(1), 23–40 (2013). DOI 10.1007/s10898-013-0089-3
38. Paulavičius, R., Žilinskas, J., Grothey, A.: Investigation of selection strategies in branch and bound algorithm with simplicial partitions and combination of Lipschitz bounds. *Optimization Letters* **4**(2), 173–183 (2010). DOI 10.1007/s11590-009-0156-3
39. Pintér, J.D.: Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications, *Nonconvex Optimization and Its Applications*, vol. 6. Springer US (1996). DOI 10.1007/978-1-4757-2502-5
40. 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
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 (2011)
42. Shi, H.J.M., Xuan, M.Q., Oztoprak, F., Nocedal, J.: On the numerical performance of derivative-free optimization methods based on finite-difference approximations (2021). DOI 10.48550/ARXIV.2102.09762. URL <https://arxiv.org/abs/2102.09762>
43. Stripinis, L., Kúdelá, J., Paulavičius, R.: Directgolib - direct global optimization test problems library (2023). URL <https://github.com/blockchain-group/DIRECTGOLib>. Pre-release v2.0
44. Stripinis, L., Paulavičius, R.: DIRECTGO: A new DIRECT-type MATLAB toolbox for derivative-free global optimization (2022). URL <https://github.com/blockchain-group/DIRECTGO>
45. Stripinis, L., Paulavičius, R.: An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms (2022). DOI 10.48550/ARXIV.2209.05759. URL <https://arxiv.org/abs/2209.05759>
46. 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
47. 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>
48. Stripinis, L., Paulavičius, R.: Directgo: A new direct-type matlab toolbox for derivative-free global optimization. *ACM Trans. Math. Softw.* **48**(4) (2022). DOI 10.1145/3559755. URL <https://doi.org/10.1145/3559755>
49. Stripinis, L., Paulavičius, R.: DIRECTGOLib - DIRECT Global Optimization test problems Library (2022). DOI 10.5281/zenodo.6617799. URL <https://doi.org/10.5281/zenodo.6617799>
50. Stripinis, L., Paulavičius, R.: Experimental study of excessive local refinement reduction techniques for global optimization direct-type algorithms. *Mathematics* **10**(20) (2022). DOI 10.3390/math10203760. URL <https://www.mdpi.com/2227-7390/10/20/3760>
51. 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>
52. Stripinis, L., Paulavičius, R.: Lipschitz-inspired halrect algorithm for derivative-free global optimization. *Journal of Global Optimization* (2023). DOI 10.1007/s10898-023-01296-7. URL <https://doi.org/10.1007/s10898-023-01296-7>
53. Surjanovic, S., Bingham, D.: Virtual library of simulation experiments: Test functions and datasets. <http://www.sfu.ca/~ssurjano/index.html> (2013). Online; accessed: 2017-05
54. Wolpert, D.H., Macready, W.G., et al.: No free lunch theorems for search. *Tech. rep.*, Citeseer (1995)
55. Wu, G., Mallipeddi, R., Suganthan, P.N.: Problem definitions and evaluation criteria for the cec 2017 competition on constrained real-parameter optimization. National University of Defense Technology, Changsha, Hunan, PR China and Kyungpook National University, Daegu, South Korea and Nanyang Technological University, Singapore, Technical Report (2017)