

Diagonal Partitioning Strategy Using Bisection of Rectangles and a Novel Sampling Scheme

Nabila Guessoum, Lakhdar Chiter✉

Department of Mathematics, Ferhat-Abbas University of Sétif1, Sétif 19000, Algeria.
Fundamental and Numerical Mathematics Laboratory.
nabila.guessoum@univ-setif.dz, lchiter@univ-setif.dz✉

Abstract

In this paper, we consider a global optimization problem where the objective function is assumed to be Lipschitz-continuous with an unknown Lipschitz constant. Building upon the recently introduced BIRECT (Bisection of RECTangles) algorithm, we propose a new diagonal partitioning and sampling scheme. Our framework, named BIRECT-V (V for vertices), combines bisection with the sampling of two points. In the initial hyper-rectangle, these points are located at $1/3$ and 1 along the main diagonal. Unlike most DIRECT-type algorithms, where evaluating the objective function at vertices is not suitable for bisection, our strategy, when combined with bisection, provides more comprehensive information about the objective function. However, the creation of new sampling points may coincide with existing ones at shared vertices, resulting in additional evaluations of the objective function and increasing the number of function evaluations per iteration. To overcome this issue, we propose modifying the original optimization domain to obtain a good approximation of the global solution. Experimental investigations demonstrate that this modification positively impacts the performance of the BIRECT-V algorithm. Our proposal shows promise as a global optimization algorithm compared to the original BIRECT and two popular DIRECT-type algorithms on a set of test problems. It particularly excels at high-dimensional problems.

Keywords: Global Optimization, BIRECT Algorithm, Diagonal Partitioning Strategy, Sampling Scheme.

Received: 21 August 2023
Accepted: 15 September 2023
Online: 18 September 2023
Published: 20 December 2023

1 Introduction

Global optimization methods have long had a prominent position in many fields. They are becoming more popular tools due to the variety and nature of the problems they may be utilized to solve. According to the method used to find the optimum, global optimization approaches can generally be divided into two major categories: deterministic [10, 11, 40, 6] and stochastic methods [19, 56]. In black-box optimization cases, the development of derivative-free global optimization algorithms has been forced by the need to optimize various and often increasingly complex problems in practice because analytic information about the objective function is unavailable.

In this paper, we consider the global optimization problem of the form

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

where the feasible domain is an n -dimensional hyper-rectangle $D = [\mathbf{a}, \mathbf{b}] = \{\mathbf{x} \in \mathbb{R}^n : a_j \leq x_j \leq b_j, j = 1, \dots, n\}$, $\mathbf{a}, \mathbf{b} \in \mathbb{R}^n$ and the objective function $f(\mathbf{x})$ is usually assumed to be Lipschitzian with maybe unknown Lipschitz constant L , $0 < L < \infty$, i.e.,

$$|f(\mathbf{x}) - f(\mathbf{y})| \leq L \|\mathbf{x} - \mathbf{y}\|, \quad \mathbf{x}, \mathbf{y} \in D. \quad (2)$$

The norm $\|\cdot\|$ denotes usually the Euclidean norm, but other equivalent norms can also be used [1, 29, 30]. The function $f(\mathbf{x})$ is also supposed to be non-differentiable; therefore, numerical methods using gradient information cannot be used to solve this kind of problem.

Various methods have been proposed to solve the optimization problem (1)-(2) using different domain partition schemes (see [10, 36, 56]). In global optimization, a feasible domain is usually a hyper-rectangle; therefore, most DIRECT-type methods use hyper-rectangular partitions. However, other types of sampling and partitioning schemes may be appropriate to some optimization problems, e.g., simplicial partitioning based on one-dimensional trisection or bisection and sampling at the center (DISIMPL-C[31]) or vertices (DISIMPL-V[30]). Other diagonal sampling schemes ([36, 37, 39, 34, 35]) use two points per hyper-rectangle instead of one point; e.g., adaptive diagonal curves (ADC algorithm[37]) use hyper-rectangular partitioning based on one-dimensional trisection and evaluate the objective function at two vertices of the main diagonals. A detailed review of different sampling and partitioning schemes is summarized in [48] and the references given therein.

DIRECT (DIvide RECTangles) algorithm developed by Jones [14, 12] is one of the most widely used partitioning-based algorithms due to its simplicity, and it only needs one algorithmic parameter ([8, 7, 4, 5, 3, 1]). The algorithm is an extension of classical Lipschitz optimization (see, e.g., [29, 2, 32, 38, 41]), where the need to know the Lipschitz constant is eliminated. However, DIRECT algorithm may converge slowly if it gets close to the optimum, requiring it to divide incessantly near the location of this optimum. The reason is that hyper-rectangles that are not potentially optimal (having bad function values at their centers) but may contain better function values will be selected in the next iterations. This procedure influences the selection of potentially optimal hyper-rectangles (having better function values), which need to be selected first.

Since its introduction, various modifications have been introduced to improve the performance of DIRECT [4, 5, 8, 7, 12, 13, 14, 3, 21, 20, 22, 23, 24, 25]. Recently, various DIRECT-type extensions and modifications have been proposed, aiming to improve the selection of potential optimal hyper-rectangles or by using different partitioning techniques leading to even more effective DIRECT-type algorithms [51, 52, 50, 42, 43, 48, 46, 44]. The recent papers [47, 15, 16] provide a good and comprehensive review of techniques in DIRECT-type algorithms. It significantly contributes to the field of derivative-free global optimization and serves as a valuable resource for researchers and practitioners seeking to enhance the efficiency and effectiveness of such algorithms.

Contrary to the most DIRECT-type algorithms, which use a central sampling strategy, the use of two points instead of one point in the sampling process, as in many diagonal-type algorithms, may reduce the probability for a hyper-rectangle with the global minimum to have a bad function value since the two (the good point and the bad function value) are in the same hyper-rectangle.

BIRECT (BIsection of RECTangles) algorithm was initially developed by Paulavičius et al. [27]. The algorithm samples two points (located at $1/3$ and $2/3$) along a diagonal per hyper-rectangle and uses bisection instead of trisection. Many arguments revealed, in a recent review [13, 26], that BIRECT gives very promising results compared to other DIRECT-type algorithms.

Since the original BIRECT algorithm was introduced, the authors in [28] suggested two-phase globally-biased extensions from [30] to the BIRECT algorithm called Gb-BIRECT, and a hybridized BIRECT algorithm Gb-BIRMIN is constructed by combining the globally-biased framework and the local optimization. They also developed in [28] a version of BIRECT called BIRECT-1 which differs from BIRECT in that only one hyper-rectangle is selected, even if several hyper-rectangles are potentially optimal.

This paper introduces a variant of the original BIRECT by modifying the location of the sampling points. Each hyper-rectangle is described by two sam-

pling points, whose positions on the corresponding diagonal are located at one-third and at the opposite farthest vertex.

In contrast to the most DIRECT-type algorithms, where the evaluation of the objective function at vertices is not favorable for bisection, this sampling strategy, combined with bisection, provides a better approximation of the objective function than central-sampling methods. Nevertheless, it is observed that the objective function could be re-evaluated more than twice at some shared vertices, leading to a significant increase in function evaluations. This strategy is typical for diagonal-based algorithms, which produce many unnecessary sampling points of the objective function. Every vertex where the function has been evaluated can belong to up to 2^n hyper-rectangles [35, 38, 18]. Especially the algorithm takes significantly longer than usual to find a solution close to a global optimum.

One of the possible suggestions to overcome these drawbacks is to consider a particular vertex database to avoid re-evaluation of the objective function [37]. The function is evaluated at every vertex only once, and then the result is directly retrieved from the database when required [35, 38, 18]. The second alternative is to group more hyper-rectangles having approximately the same size in the same group, which effectively reduces the set of selected potentially optimal hyper-rectangles. Some suggested methods are summarized in [51, 48, 28, 13]. This possibility is not considered in the present paper but can be observed, for example, in the case of BIRECT-V1, since it selects only one potentially optimal hyper-rectangle from each group. This situation is favorable, especially when a good objective function value attained at the vertex can belong to many (up to 2^n) hyper-rectangles. One possibility is to consider an appropriate (tight) Lipschitzian lower bound, since we observed that Eq. 5 is much more in favor of BIRECT than of BIRECT-V. Another alternative that seems very attractive is to modify the original optimization domain for some test problems to obtain a good approximation to the global solution. The influence of such a modification on the performance of the BIRECT-V algorithm is as efficient as the number of function evaluations required to get close to a good solution.

It is clear that this last alternative does not overcome the situation in a proper way, but at least it helps to reduce considerably the number of function evaluations.

Therefore, the main purpose of this paper is to focus on this particular scheme (sampling at vertices) without any additional parameters to the BIRECT-V algorithm framework by investigating this new approach and discussing its advantages and drawbacks.

Consequently, the contribution of this paper is summarized as follows:

- A new modified BIRECT algorithm is suggested, named BIRECT-V.
- A new variation of the BIRECT-V algorithm, called BIRECT-V1 is also introduced.

- The new approach incorporates bisection with sampling on diagonal vertices, which is not commonly used in the majority of existing BIRECT-type algorithms.
- Numerical Comparison on Test Problems shows the advantages of the approach.
- It is shown that a modification in the original domain can have a positive impact on the performance of the BIRECT-V algorithm.
- An innovative extension built upon our approach to handle global optimization problems involving Lipschitz continuous functions subject to linear constraints seems to address a challenging and relevant problem [49].

The remainder of this paper is organized as follows: In Sect. 2, we outline the working principles of the original BIRECT. This will make more comprehensible the ideas behind the BIRECT-V algorithm to be proposed. A description of the new sampling and partitioning scheme is given in Sect. 2.2. Implementation of the BIRECT-V algorithm along with the other DIRECTtype algorithms is given in Sect. 3. Numerical investigation and comparison with BIRECT, BIRECT-1, and two DIRECTtype algorithms on 54 variants of Hedar test problems [9] is presented in Sect. 3.2. Finally, in Sect. 4, we conclude the paper with some remarks and directions for future research.

2 Materials and Methods

In this section we start by giving a description of the principle of the sampling and division strategies retained from the original BIRECT algorithm. Then we introduce our suggested method with emphasis on the sampling strategy. We conclude this section by an illustration of this new scheme.

2.1 From BIRECT to BIRECT-V

The original BIRECT (BIsection of RECTangles) algorithm, developed by Paulavičius et al.[27], is based on a diagonal space-partitioning technique and includes two main procedures: sampling on diagonals and using bisection of hyper-rectangles. The algorithm begins by scaling the initial search space D to the unit hyper-cube \bar{D} , where all the variables are returned. At the initialization step of BIRECT, $f(\mathbf{x})$ is evaluated at two points "lower" $\mathbf{l} = (l_1, \dots, l_n) = (1/3, \dots, 1/3)^T$ and "upper" $\mathbf{u} = (u_1, \dots, u_n) = (2/3, \dots, 2/3)^T$ located on the main diagonal of the normalized domain \bar{D} , equidistant between themselves and the endpoints of the diagonal. The hyper-cube is then partitioned into a set of smaller hyper-rectangles and $f(\mathbf{x})$ is evaluated over each hyper-rectangle at two diagonal points by following a specific sampling and partitioning scheme obeying the two following rules.

2.1.1 Selection Rule

Let the partition of \bar{D} at iteration k be defined as

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

where $\bar{D}^i = [\mathbf{a}^i, \mathbf{b}^i] = \{\mathbf{x} \in \mathbb{R}^n : l^i \leq \mathbf{x} \leq u^i, \forall i \in \mathbb{I}_k\}$, $l^i, u^i \in [0, 1]$ and \mathbb{I}_k is the set of indices identifying the subsets defining the current partition \mathcal{P}_k . At the generic k th iteration, starting from the current partition \mathcal{P}_k of \bar{D}^i , a new partition \mathcal{P}_{k+1} is obtained by bisecting a set of *potentially optimal hyper-rectangles* from the previous partition \mathcal{P}_k . The identification of a potentially optimal hyper-rectangle is based on the lower bound estimates for $f(\mathbf{x})$ over each hyper-rectangle by fixing some *rate of change* $\tilde{L} > 0$ (which has a role analogous to a Lipschitz constant). A hyper-rectangle $\bar{D}^j, j \in \mathbb{I}_k$. We call *potentially optimal* a hyper-rectangle j if $\forall i \in \mathbb{I}_k$, the following inequalities hold

$$\begin{aligned} \min \{f(\mathbf{l}^j), f(\mathbf{u}^j)\} - \tilde{L}\delta_j &\leq \min \{f(\mathbf{l}^i), f(\mathbf{u}^i)\} - \tilde{L}\delta_i, (3) \\ \min \{f(\mathbf{l}^j), f(\mathbf{u}^j)\} - \tilde{L}\delta_j &\leq f_{\min} - \varepsilon|f_{\min}|, (4) \end{aligned}$$

where the measure (distance, size) of the hyper-rectangle is given by

$$\delta_i = \frac{2}{3} \|\mathbf{b}^i - \mathbf{a}^i\|, \quad (5)$$

$\varepsilon > 0$ is a positive constant, and f_{\min} is the current best known function value. A hyper-rectangle j is *potentially optimal* if the lower bound for f computed by the left-hand side of (3) is optimal for some fixed rate of change \tilde{L} among the hyper-rectangles of the current partition \mathcal{P}_k . Inequality (4) ensures guarding against an excessive emphasis on the local search [14].

2.1.2 Division and Sampling Rule

After the initial covering, BIRECT-V moves to the future iterations by partitioning *potentially optimal hyper-rectangles* and evaluating the objective function $f(x)$ at their new sampling points.

New sampling points are obtained by adding and subtracting from the previous (old) ones a distance equal to the half-side length of the branching coordinate. This way, old sampled from the previous iterations are re-used in descendant subregions.

A vital aspect of the algorithm is how the selected hyper-rectangles $\bar{D}^i, i \in \mathbb{I}_k$ are divided. For every *potentially optimal* hyper-rectangle the set of the maximum coordinates (edges) is computed, and every *potentially optimal* hyper-rectangle is bisected (divided in halves of equal size), along the coordinate (branching variable $x_{br}, 1 \leq br \leq n$), having the largest side length (d_{br}^i) and by first considering the coordinate directions with the lowest index j (if more coordinates may be chosen), where function values are more promising, [55].

$$br = \min \left\{ \arg \max_{1 \leq j \leq n} = \{d_j^i = |b_j^i - a_j^i|\} \right\}, \quad (6)$$

The partitioning process continues until a prescribed number of function evaluations has been performed, or a stopping criterion is satisfied. The best (smaller) found objective function value $f(\bar{\mathbf{x}})$ over all sampled points of the final partition, and the corresponding generated point $\bar{\mathbf{x}}$, provide an approximate solution to the problem.

Further details and comprehensive description of the original BIRECT algorithm can be found in Paulavicius et al.[27].

2.2 Description of the New Sampling Scheme

In this subsection, we present the basic idea of the new sampling scheme in a more general setting. An illustration is given in a two-dimensional example in Fig. 1 and Fig. 2. Since our new method is based on the original BIRECT algorithm, BIRECT-V follows the same hyper-rectangle selection and subdivision procedure, unlike the sampling method which is done in a different way.

In the initialization phase, BIRECT-V normalize the search domain to an n -dimensional unit hyper-rectangle \bar{D}_0^1 , and evaluates the objective function $f(\mathbf{x})$ at two different diagonal points: "third" $\mathbf{t}^i = (t_1^i, \dots, t_n^i) = (1/3, \dots, 1/3)^T$ and "vertex" $\mathbf{v}^i = (v_1^i, \dots, v_n^i) = (1, \dots, 1)^T$. The scaled hyper-rectangle is considered as the only trivial selected POH.

In the succeeding iterations, POHs are selected and bisected in essentially the same way as BIRECT, with the change that in inequalities (3) and (4), the sampled points \mathbf{l}^i and \mathbf{u}^i are replaced by $\mathbf{t}^i = \mathbf{l}^i$ and $\mathbf{v}^i = \mathbf{u}^i + \frac{1}{3}\|\mathbf{b}^i - \mathbf{a}^i\|$ respectively, and using the same measure of the hyper-rectangle given by Eq. (5).

Selected POHs are divided with the restriction that only along the coordinate (branching variable x_{br} , $1 \leq br \leq n$), having the largest side length (d_{br}^i), and by first considering the coordinate directions with the smallest index j (if more coordinates may be chosen). This restriction guarantees that the hyper-rectangle will reduce on every dimension. Potentially optimal hyper-rectangles are shown in the left-side of Fig. 3, and correspond to the lower-right convex hull of the set of points.

Formalizing our sampling and partitioning schemes in a more general case. Suppose that at iteration k , $\bar{D}_k^i = [\mathbf{a}^i, \mathbf{b}^i] = \{\mathbf{x} \in \bar{D} : 0 \leq a_j^i \leq x_j \leq b_j^i \leq 1, j = 1, \dots, n, \forall i \in \mathbb{I}_k\}$ is a hyper-cube.

Since all the variables ($x_j, j = 1, \dots, n$) of \bar{D}_k^i have the same side lengths ($d_j^i = |b_j^i - a_j^i|, j = 1, \dots, n$), \bar{D}_k^i is bisected (divided in halves) across the middle point $\frac{1}{2}(a_1^i + b_1^i)$ of the coordinate direction with the smallest index ($x_j, j = 1$) into two hyper-rectangles \bar{D}_k^{i+1} , and \bar{D}_k^{i+2} of equal side lengths (see Fig. 1, iteration 1 for illustration).

After \bar{D}_k^i is bisected, the first iteration is performed by sampling two new points from the old ones.

The new point \mathbf{t}^{i+2} is obtained by adding or subtracting from the old point one third side-length $d_{br}^i/3$

to the lower coordinate of the branching variable. Also the new point \mathbf{v}^{i+1} is obtained from the old one by subtracting or adding the whole side length d_{br}^i , while keeping all the rest of coordinates issued from \mathbf{t}^i and \mathbf{v}^i unchanged.

In the case where \bar{D}_k^i is a hyper-rectangle, new sampled points are obtained, after distinguishing the branching variable (br), by adding or subtracting the required side length from the coordinate on which we branch, pursuant the following rule:

If $t_j^i < v_j^i$, then

$$t_{br}^{i+2} = t_{br}^i + \frac{d_{br}^i}{3}, \quad \text{and} \quad v_{br}^{i+1} = v_{br}^i - d_{br}^i, \quad (7)$$

otherwise, i.e., if $t_j^i > v_j^i$, then

$$t_{br}^{i+1} = t_{br}^i - \frac{d_{br}^i}{3}, \quad \text{and} \quad v_{br}^{i+2} = v_{br}^i + d_{br}^i. \quad (8)$$

The two new points are obtained as follows:

$$\mathbf{t}^{i+2} = (t_1^i, \dots, t_{br}^i \pm \frac{d_{br}^i}{3}, \dots, t_n^i) = (t_1^i, \dots, t_{br}^i \pm \frac{|b_1^i - a_1^i|}{3}, \dots, t_n^i),$$

and

$$\mathbf{v}^{i+1} = (v_1^i, \dots, v_{br}^i \pm d_{br}^i, \dots, v_n^i) = (v_1^i, \dots, v_{br}^i \pm |b_1^i - a_1^i|, \dots, v_n^i).$$

Each descending hyper-rectangle \bar{D}_k^{i+1} and \bar{D}_k^{i+2} retains one sampled point \mathbf{t}^i and \mathbf{v}^i , respectively from their ancestor \bar{D}_k^i . At the same time, old sampling points are re-used in descending hyper-rectangles as $\mathbf{t}^{i+1} = \mathbf{t}^i$ and $\mathbf{v}^{i+2} = \mathbf{v}^i$. More precisely:

$$\begin{aligned} \mathbf{t}^{i+1} &= \mathbf{t}^i = (t_1^i, \dots, t_n^i) \\ &= (a_1^i + \frac{1}{3}|b_1^i - a_1^i|, \dots, a_n^i + \frac{1}{3}|b_n^i - a_n^i|) \\ &= (a_1^{i+1} + \frac{2}{3}|b_1^{i+1} - a_1^{i+1}|, \dots, a_n^{i+1} + \frac{1}{3}|b_n^{i+1} - a_n^{i+1}|), \end{aligned}$$

and

$$\begin{aligned} \mathbf{v}^{i+2} &= \mathbf{v}^i = (v_1^i, \dots, v_n^i) \\ &= (a_1^i + |b_1^i - a_1^i|, \dots, a_n^i + |b_n^i - a_n^i|) \\ &= (a_1^{i+2} + |b_1^{i+2} - a_1^{i+2}|, \dots, a_n^{i+2} + |b_n^{i+2} - a_n^{i+2}|). \end{aligned}$$

The BIRECT-V algorithm continues in this way by sampling two new points in each potentially optimal hyper-rectangle, by adding and subtracting the required side-length from the old points, and bisecting through the longest coordinate until some stopping rule is satisfied. After subdivision, each rectangle resulting from the previous iteration retains one point from its predecessor.

Notice that the sampled points \mathbf{v}^{i+1} and \mathbf{v}^{i+1} in \bar{D}_k^{i+1} belong to the same diagonal (see Fig. 1 for illustration). This is a straightforward consequence of Theorem 1 in [27]. The same conclusion holds for hyper-rectangle \bar{D}_k^{i+2} .

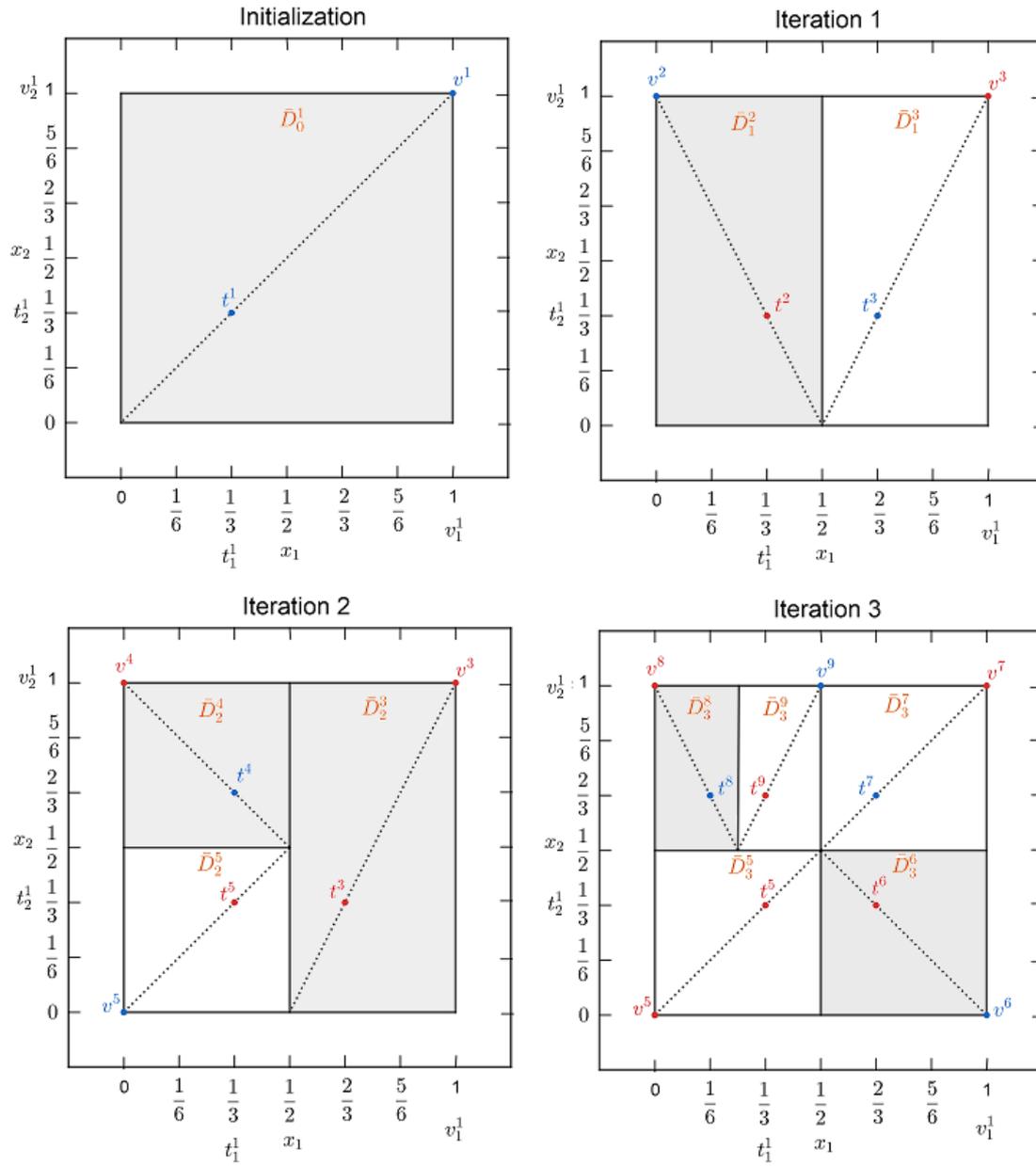


Figure 1: Description of the initialization and the first three iterations used in the new sampling scheme on the Branin test problem. Each iteration is performed by sampling two new points (blue color) issued from the old ones (red color) and bisecting potentially optimal hyper-rectangles (shown in gray color) along the coordinate (branching variable x_{br} , $1 \leq br \leq n$), having the largest side length (d_{br}^i , where $d_j^i = |b_j^i - a_j^i|$, $j = 1, \dots, n$) and by first considering the coordinate directions with the smallest index j (if more coordinates may be chosen).

Finally, let us emphasize that, in contrast to the naming convention used in [27] of the sampling points as “lower” (l) and “upper” (u), to make differentiate two points belonging to the same hyper-rectangle, we can assume without any confusion that the new points are affected as “third” t and “vertex” v . In this way, the two points are always identified during all the op-

timization process even if they are “lower” or “upper”.

It is also of importance to stress again, that our new sampling scheme differs in its unique and distinctive way on how new sampled points are created by using different side-lengths, in contrast to direct-type algorithms and diagonal sampling strategies, where they use the same side-lengths.

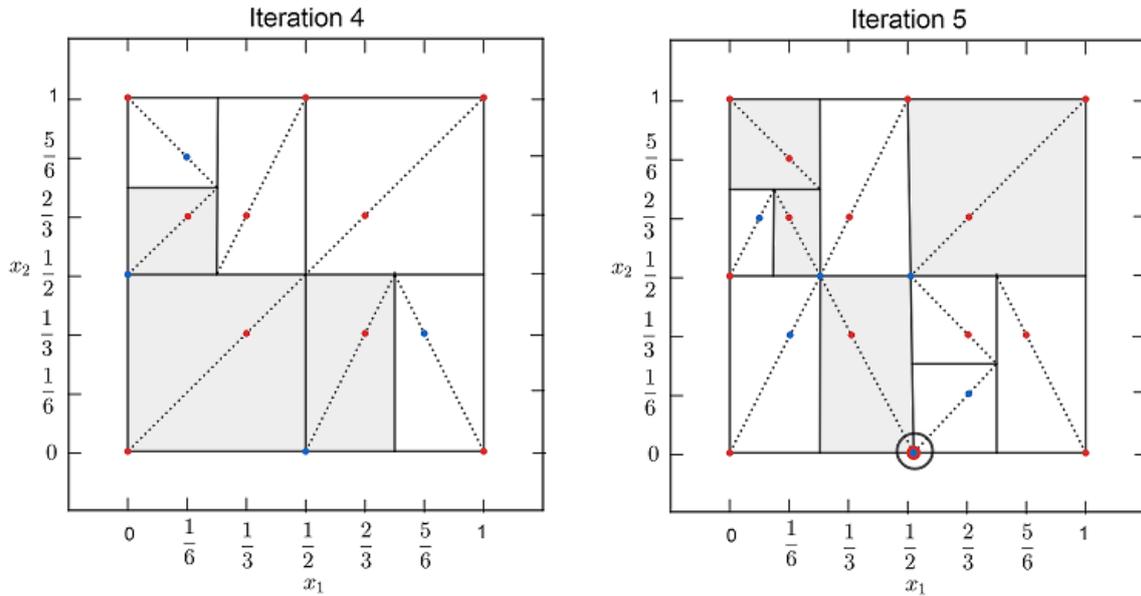


Figure 2: Illustration of selection, sampling and partitioning schemes ranging from iteration 4 to 5 on the Branin test problem. A situation where two adjacent hyper-rectangles share the same vertex. After bisection of the lower-left hyper-rectangle in iteration 4, the new created point fall exactly with the one in the adjacent hyper-rectangle. This point is marked with a circle in iteration 5.

2.2.1 Illustration

Let $\mathbf{t}^1 = (t_1^1, t_2^1) = (1/3, 1/3)$ and $\mathbf{v}^1 = (v_1^1, v_2^1) = (1, 1)^T$ denote two points lying on the main diagonal (see initialization in Fig. 1) of hyper-rectangle $D_0^1 = [\mathbf{a}^1, \mathbf{b}^1] = [a_1^1, b_1^1] \times [a_2^1, b_2^1]$.

Without losing generality, we restrict our illustration to two iterations only; the other situations are the same. In (Fig. 1, iteration 2), \bar{D}_2^3 and \bar{D}_2^4 are POHs. For hyper-rectangle \bar{D}_2^3 , as there is only one longest side (coordinate $j = 2$) with side length $d_2^3 = 1$. Therefore using the rule in Eq. 7, the new sampling points \mathbf{t}^7 and \mathbf{v}^6 are expressed as follows:

$$\mathbf{t}^7 = (t_1^7, t_2^7) = \left(t_1^3, t_2^3 + \frac{d_2^3}{3} \right) = \left(\frac{2}{3}, \frac{2}{3} \right),$$

$$\mathbf{v}^6 = (v_1^6, v_2^6) = (v_1^3, v_2^3 - d_2^3) = (1, 0).$$

For hyper-rectangle \bar{D}_2^4 , we use the second rule given by Eq. 8. The new sampling points are located at (see Fig. 1, iteration 2):

$$\mathbf{t}^8 = \left(t_1^4 - \frac{d_1^4}{3}, t_2^4 \right) = \left(t_1^4 - \frac{1}{3}, t_2^4 \right) = \left(\frac{1}{6}, \frac{2}{3} \right),$$

$$\mathbf{v}^9 = (v_1^4 + d_1^4, v_2^4) = (v_1^4 + 1, v_2^4) = \left(\frac{1}{2}, 1 \right).$$

However, in Fig. 2, we encounter a situation where two adjacent hyper-rectangles share the same vertex. After bisection of the lower-left hyper-rectangle in iteration 4, the newly created point falls exactly with

the one in the adjacent hyper-rectangle. This point is marked with a circle in iteration 4. This situation is shown on the right side of Fig. 3, where we distinguish three sampled points at which the objective function has been evaluated twice at this vertex. Such a difference becomes more pronounced as optimization proceeds.

2.2.2 Main steps of the BIRECT-V Algorithm

The BIRECT-V algorithm main steps are shown in Algorithm 1, where the inputs are problem (f), optimization domain (D), and some stopping criteria: required tolerance (ϵ_{pe}), the maximal number of function evaluations (M_{max}), and the maximal number of iterations (K_{max}). BIRECT-V returns the value of the objective function found (f_{min}), and the point (x_{min}) as well as the algorithmic performance measures: percent error (pe), number of function evaluations (m), and number of iterations (k) after termination.

The BIRECT-V algorithm begins the initialization phase by the normalization of the feasible domain (D), evaluating the objective function (f) at the two first sampling points \mathbf{t}^1 and \mathbf{v}^1 , measuring and setting stopping conditions (see Algorithm 1, lines 2-4). lines 5-21 of Algorithm 1 describes the main while loop, which is executed until one of the stopping conditions specified is met. As explained in the previous section (see Subsect. 2.2.1), the BIRECT-V algorithm, at the beginning of each iteration, identifies the set of POHs (see Algorithm 1, line 7, excluding steps 7 (high-

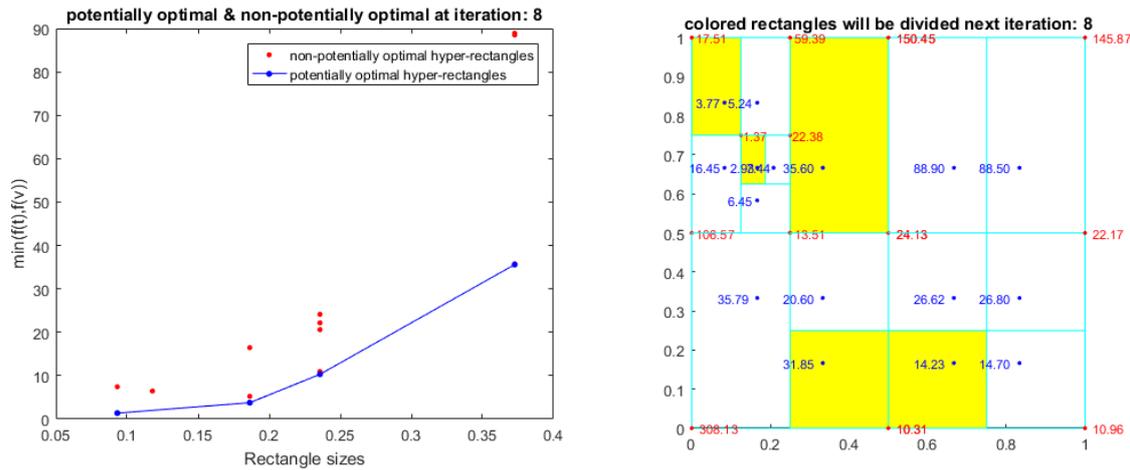


Figure 3: Geometric interpretation of potentially optimal hyper-rectangles using the BIRECT-V algorithm on the Branin test function in the seventh iteration: (*right side*), POHs correspond to the lower-right convex hull of points marked in blue color (*left side*). The position of six points (values of $f(x)$) obtained in BIRECT can be clearly distinguished. We observe three sampled points at which the objective function has been re-evaluated.

lighted in magenta color), which are performed only on the BIRECT-V1 algorithm). (see Algorithm 1, line 6), then bisects all POHs (Algorithm 1, line 11) and creates the new sampling points t^i and v^i of generated hyper-rectangles (see Algorithm 1, line 12). Finally, BIRECT-V found a solution, and the performance measures are returned (see Algorithm 1, line 22).

The structure of BIRECT-V is outlined in Algorithm 1.

2.2.3 Convergence

Since BIRECT-V is based on the ideas of BIRECT, therefore the convergence of BIRECT-V could be determined as many times as other DIRECT-type algorithms [14, 12, 8, 7], in the sense of the "everywhere-dense" type of convergence (see [33]). In addition, the continuity of the objective function in the neighborhood of global minima is a sufficient assumption that guarantees convergence.

3 Results and Discussion

This section provides a description of the experimental results, their interpretation, and the experimental conclusions. we compare the performance of our newly introduced modification, BIRECT-V, and its variant, called BIRECT-V1, which differs from BIRECT-V in that, if several rectangles are tied for being potentially optimal, only one of them is selected. with the original BIRECT algorithm, BIRECT-1 [27, 28], and two other well-known DIRECT-type algorithms [14, 12, 8, 7].

3.1 Implementation

As the BIRECT-V algorithm is based on the original BIRECT algorithm, we use the same measure of the size of the hyper-rectangle.

Note that in the DIRECT algorithm, this size is measured by the Euclidean distance from its center to a corner, while in DIRECT-1, it corresponds to the infinity norm, permitting the algorithm to collect more hyper-rectangles having the same size. In BIRECT-V1, the number of potentially hyper-rectangles in each group, to be further divided, is reduced to at most one hyper-rectangle.

In Table 1 are listed the test problems from [9] used in this comparison, which consist in total of 54 global optimization test problems with dimensions varying from $n = 2$ to $n = 10$, with the main attributes: problem number, problem name, dimension (n), feasible domain (D), number of local minima, and known minimum (f^*). Note that these problems could also be found in [53], and in a more detailed version in [50] and related up-to-date versions. In our study, we chose to benchmark using the Hedar test set [9] instead of alternatives like the BBOB set and the GKLS generator [16, 17]. This choice aligns with the specific objectives and scope of our research. It's important to emphasize that we are comparing BIRECTv here only with other "pure" DIRECT-type algorithms. Consequently, our primary objective was to explore two distinct strategies: one involving sampling and bisection techniques employed in the new modification of BIRECT, contrasted with the original version of BIRECT, one sampling commonly employed in the majority of DIRECT-type algorithms. Utilizing multiple benchmark sets, we ensure a comprehensive analysis of our method's performance across various problem domains. This choice allows us to assess our method's generalizability and robustness by evaluating its performance on a diverse set of optimization problems.

Some of these test problems have several variants, e.g. (*Bohachevsky, Hartmann, Shekel*), while others (*Ackley, Dixon and Price, Levy, Rastrigin, Rosenbrock,*

BIRECT-VI				BIRECT-V			
iteration: 1	fmin: 4.6082847879	f evals: 2	.	iteration: 1	fmin: 4.6082847879	f evals: 2	.
.
iteration: 50	fmin: 3.2479917988	f evals: 12	.	iteration: 50	fmin: 3.2479917988	f evals: 522	.
.	.	.	.	iteration: 133	fmin: 0.0042301342	f evals: 2028	.
iteration: 150	fmin: 0.0007342074	f evals: 14	.	iteration: 134	fmin: 0.0040898808	f evals: 1294	.
.	.	.	.	iteration: 135	fmin: 0.0039448443	f evals: 2422	.
iteration: 170	fmin: 0.0002239623	f evals: 4	.	iteration: 136	fmin: 0.0037944837	f evals: 2482	.
.
iteration: 188	fmin: 0.0002239623	f evals: 8	.	iteration: 150	fmin: 0.0007342074	f evals: 1746	.
iteration: 189	fmin: 0.0002225978	f evals: 10	.	iteration: 189	fmin: 0.0002225978	f evals: 2430	.
iteration: 190	fmin: 0.0000152596	f evals: 10	.	iteration: 190	fmin: 0.0000152596	f evals: 3306	.

Figure 4: Iteration progress of the BIRECT-VI algorithm on the left-hand side and BIRECT-V on the right-hand side while solving the Ackley (No. 3, $n=10$, from Table 5) test problem.

Table 1: Key characteristics of the Hedar test problems[9].

Problem No.	Problem name	Dimension n	Feasible region $D = ([a_j, b_j], j = 1, \dots, n)$	No. of local minima	Optimum f^*
1*, 2*, 3*	Ackley	2, 5, 10	$[-15, 35]^n$	multimodal	0.0
4	Beale	2	$[-4.5, 4.5]^2$	multimodal	0.0
5*	Bohachevsky 1	2	$[-100, 110]^2$	multimodal	0.0
6*	Bohachevsky 2	2	$[-100, 110]^2$	multimodal	0.0
7*	Bohachevsky 3	2	$[-100, 110]^2$	multimodal	0.0
8	Booth	2	$[-10, 10]^2$	unimodal	0.0
9	Branin	2	$[-5, 10] \times [10, 15]$	3	0.39789
10	Colville	4	$[-10, 10]^4$	multimodal	0.0
11, 12, 13	Dixon & Price	2, 5, 10	$[-10, 10]^n$	unimodal	0.0
14	Easom	2	$[-100, 100]^2$	multimodal	-1.0
15	Goldstein & Price	2	$[-2, 2]^2$	4	3.0
16*	Griewank	2	$[-600, 700]^2$	multimodal	0.0
17	Hartman	3	$[0, 1]^3$	4	-3.86278
18	Hartman	6	$[0, 1]^6$	4	-3.32237
19	Hump	2	$[-5, 5]^2$	6	-1.03163
20, 21, 22	Levy	2, 5, 10	$[-10, 10]^n$	multimodal	0.0
23*	Matyas	2	$[-10, 15]^2$	unimodal	0.0
24	Michalewics	2	$[0, \pi]^2$	2!	-1.80130
25	Michalewics	5	$[0, \pi]^5$	5!	-4.68765
26	Michalewics	10	$[0, \pi]^{10}$	10!	-9.66015
27	Perm	4	$[-4, 4]^4$	multimodal	0.0
28, 29	Powell	4, 8	$[-4, 5]^n$	multimodal	0.0
30	Power Sum	4	$[0, 4]^4$	multimodal	0.0
31*, 32*, 33*	Rastrigin	2, 5, 10	$[-5.12, 6.12]^n$	multimodal	0.0
34, 35, 36	Rosenbrock	2, 5, 10	$[-5, 10]^n$	unimodal	0.0
37, 38, 39*	Schwefel	2, 5, 10	$[-500, 500]^n$	unimodal	0.0
40	Shekel, $m = 5$	4	$[0, 10]^4$	5	-10.15320
41	Shekel, $m = 7$	4	$[0, 10]^4$	7	-10.40294
42	Shekel, $m = 10$	4	$[0, 10]^4$	10	-10.53641
43	Shubert	2	$[-10, 10]^2$	760	-186.73091
44*, 45*, 46*	Sphere	2, 5, 10	$[-5.12, 6.12]^n$	multimodal	0.0
47*, 48*, 49*	Sum squares	2, 5, 10	$[-10, 15]^n$	unimodal	0.0
50	Trid	6	$[-36, 36]^6$	multimodal	-50.0
51	Trid	10	$[-100, 100]^{10}$	multimodal	-210.0
52*, 53*, 54*	Zakharov	2, 5, 10	$[-5, 11]^n$	multimodal	0.0

Table 2: Preliminary results during the first run of the BIRECT-V algorithm.

Problem No.	Optimum f^*	BIRECT-V1		BIRECT-V	
		$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.
1	0.0	7.42×10^{-5}	198	7.42×10^{-5}	342
2	0.0	9.17×10^{-5}	422	9.17×10^{-5}	3514
3	0.0	9.69×10^{-5}	984	9.69×10^{-5}	70690
4	0.0	8.77×10^{-5}	640	8.77×10^{-5}	1034
5	0.0	7.14×10^{-5}	676	7.14×10^{-5}	656
6	0.0	5.96×10^{-5}	692	5.96×10^{-5}	694
7	0.0	7.58×10^{-5}	902	7.58×10^{-5}	1062
8	0.0	6.10×10^{-5}	234	6.10×10^{-5}	254
9	0.39789	0.39790	656	0.39790	492
10	0.0	9.82×10^{-5}	2320	9.82×10^{-5}	1910
11	0.0	8.92×10^{-5}	940	8.92×10^{-5}	1432
12	0.0	9.34×10^{-5}	28034	9.36×10^{-5}	23412
13	0.0	8.79×10^{-3}	> 500000	4.73×10^{-4}	> 500000
14	-1.0	-0.99999	180	-0.99999	1082
15	3.0	3.00000	28	3.00000	28
16	0.0	5.13×10^{-5}	8288	5.13×10^{-5}	8950
17	-3.86278	-3.86244	200	-3.86244	208
18	-3.32237	-3.32214	542	-3.32214	542
19	-1.03163	-1.03154	202	-1.03154	334
20	0.0	1.44×10^{-5}	188	1.44×10^{-5}	226
21	0.0	7.56×10^{-5}	674	7.56×10^{-5}	1000
22	0.0	9.27×10^{-5}	2082	9.27×10^{-5}	18676
23	0.0	2.71×10^{-5}	148	2.71×10^{-5}	208
24	-1.80130	-1.80130	184	-1.80130	314
25	-4.68736	-4.64588	> 500000	-4.68732	339818
26	-9.66015	-8.60560	> 500000	-7.55568	> 500000
27	0.0	0.00000	80890	0.00000	62368
28	0.0	4.59×10^{-5}	2786	4.59×10^{-5}	1678
29	0.0	9.15×10^{-5}	387440	9.15×10^{-5}	467200
30	0.0	0.00000	204	0.00000	204
31	0.0	0.00000	14	0.00000	16
32	0.0	0.00000	204	0.00000	210
33	0.0	0.00000	14360	0.00000	14348
34	0.0	9.65×10^{-5}	698	9.65×10^{-5}	718
35	0.0	2.41×10^{-5}	2444	2.41×10^{-5}	2972
36	0.0	5.42×10^{-5}	16506	5.42×10^{-5}	39846
37	0.0	5.64×10^{-5}	446	5.64×10^{-5}	580
38	0.0	9.49×10^{-5}	63908	9.49×10^{-5}	23022
39	0.0	1.79×10^{-8}	2938	3.42×10^{-5}	134562
40	-10.15320	-10.15234	6618	-10.15234	5866
41	-10.40294	-10.40201	2298	-10.40201	2604
42	-10.53641	-10.53544	2498	-10.53544	3324
43	-186.73091	-186.72944	806	-186.72944	1684
44	0.0	1.15×10^{-5}	112	1.15×10^{-5}	190
45	0.0	2.87×10^{-5}	392	2.87×10^{-5}	1400
46	0.0	5.74×10^{-5}	1054	5.74×10^{-5}	27566
47	0.0	8.74×10^{-5}	248	8.74×10^{-5}	280
48	0.0	3.97×10^{-5}	1354	3.97×10^{-5}	1776
49	0.0	9.35×10^{-5}	3394	9.35×10^{-5}	9244
50	-50.0	-49.99511	1402	-49.99511	2112
51	-210.0	-209.98223	168432	-209.98155	368312
52	0.0	0.00000	78	0.00000	78
53	0.0	0.00000	22498	0.00000	24150
54	0.0	1.13284	> 500000	1.21289	> 500000
Average		52485.148		58762.852	
Median		921.000		1681.000	

Schwefel, Sphere, Sum squares, Zakharov) and can be tested for different dimensionality.

Finally, notice that it may occur occasionally that at the initial steps of the algorithm, the sampling is performed near the global minimizer. In this particular situation, the feasible domain was modified in the same way as in [27], i.e., the upper bound was increased. For clarity, the modified test problems are marked with an asterisk.

Implementation and comparison of the newly introduced scheme with the original BIRECT together with other DIRECT-type algorithms were performed in the MATLAB programming language, using MATLAB R2016a on an EliteBook with the following hardware settings:

Intel Core i5-6300U CPU @ 2.5 GHz, 8 GB of memory, and running on the Windows 10 operating system (64-bit). Potentially optimal hyper-rectangles are identified using modified Graham’s scan algorithm. In our implementation, the output values are rounded up to 10 decimals. A test problem is considered successful if an algorithm returns a value of an objective function that does not exceed 10^{-4} error or a minimizer x_{min} that achieves a comparable value in [43].

The algorithms were stopped either when the point \bar{x} (noted also x_{min}) was generated such that the following stopping criterion was satisfied:

$$pe = \begin{cases} \frac{f(\bar{x}) - f^*}{|f^*|} \leq 10^{-4}, & f^* \neq 0, \\ f(\bar{x}) \leq 10^{-4}, & f^* = 0, \end{cases} \quad (9)$$

(where f^* is the known global optimum) or when the number of function evaluations exceeds the prescribed limit of 500,000. (The maximum number of iterations

was set to 100,000 but usually it is supposed to be unlimited.)

The comparison is based on two criteria: the best-found function value $f(\bar{x})$ and the number of function evaluations (f.eval.). For each test problem, the average and median numbers of function evaluations are shown at the bottom of each table. The best number of function evaluations is shown in bold font in Table 5. The number of iterations and the execution time (measured in seconds) are only reported in Tables 2 and 3 in the link <https://data.mendeley.com/datasets/x9fpc9w7wh>.

3.2 Discussion

In this subsection, we discuss the efficiency of the newly introduced BIRECT-V algorithm and compare it with the original BIRECT, BIRECT-1 (see [28, 27]) and two DIRECT-type algorithms. In Table 2, we report the results obtained by BIRECT-V and BIRECT-V1 when the algorithm is running in the usual way without additional parameters.

In Table 3 are reported the results when the best-found objective function value $f(\bar{x})$ found by the BIRECT algorithm is used as a known optimal (minimal) value (f^*). In Table 5, are summarized the experimental results for all tested algorithms are summarized and compared in the case where the original domain (D) was modified. Also, the results related to this comparison are presented in Table 4.

First, it is easy to observe from Table 2, that our proposed partitioning scheme requires, most often, more function evaluations than in BIRECT and BIRECT-1,

Table 3: BIRECT-V1 and BIRECT-V versus BIRECT and BIRECT-1.

Problem No.	Optimum f^*	BIRECT-V1		BIRECT-V	
		$f(\bar{x})$	f.eval.	$f(\bar{x})$	f.eval.
1	0.0000000e+00	2.54×10^{-5}	206	2.54×10^{-5}	360
2	0.0000000e+00	2.54×10^{-5}	438	2.54×10^{-5}	4196
3	0.0000000e+00	2.54×10^{-5}	1020	2.54×10^{-5}	73452
4	0.0000000e+00	8.77×10^{-5}	640	8.77×10^{-5}	1034
5	0.0000000e+00	4.02×10^{-5}	1078	4.02×10^{-5}	1040
6	0.0000000e+00	2.19×10^{-5}	1138	2.19×10^{-5}	1122
7	0.0000000e+00	3.67×10^{-5}	932	3.67×10^{-5}	1106
8	0.0000000e+00	3.81×10^{-6}	364	3.81×10^{-6}	376
9	$3.9789000e-01$	0.39790	656	0.39790	492
10	0.0000000e+00	4.36×10^{-5}	2568	4.36×10^{-5}	2182
11	0.0000000e+00	4.84×10^{-5}	1268	3.31×10^{-5}	1472
12	0.0000000e+00	5.99×10^{-5}	28368	4.78×10^{-5}	23902
13	0.0000000e+00	8.79×10^{-3}	> 500000	4.73×10^{-4}	> 500000
14	-1.0000000e+00	-0.99999	180	-0.99999	1082
15	3.0000000e+00	3.00000	28	3.00000	28
16	0.0000000e+00	4.61×10^{-7}	8456	4.61×10^{-7}	9162
17	-3.8627800e+00	-3.86244	200	-3.86244	208
18	-3.3223700e+00	-3.32214	542	-3.32214	542
19	-1.0316300e+00	-1.03152	168	-1.03152	274
20	0.0000000e+00	1.44×10^{-5}	188	1.44×10^{-5}	226
21	0.0000000e+00	1.12×10^{-5}	870	1.12×10^{-5}	1406
22	0.0000000e+00	2.84×10^{-5}	2642	2.84×10^{-5}	24978
23	0.0000000e+00	1.70×10^{-6}	244	1.70×10^{-6}	318
24	-1.8013000e+00	-1.80130	184	-1.80130	314
25	-4.6873600e+00	-4.645885	> 500000	-4.68732	339818
26	-9.6601500e+00	-7.452392	646	-7.37292	2408
27	0.0000000e+00	0.00000	80890	0.00000	62368
28	0.0000000e+00	4.59×10^{-5}	2786	4.59×10^{-5}	1678
29	0.0000000e+00	9.15×10^{-5}	387440	9.15×10^{-5}	467200
30	0.0000000e+00	0.00000	204	0.00000	204
31	0.0000000e+00	0.00000	14	0.00000	16
32	0.0000000e+00	0.00000	204	0.00000	210
33	0.0000000e+00	0.00000	14360	0.00000	14348
34	0.0000000e+00	9.65×10^{-5}	698	9.65×10^{-5}	718
35	0.0000000e+00	2.41×10^{-5}	2444	2.41×10^{-5}	2972
36	0.0000000e+00	5.42×10^{-5}	16506	5.42×10^{-5}	39846
37	0.0000000e+00	5.64×10^{-5}	446	5.62×10^{-5}	580
38	0.0000000e+00	6.41×10^{-5}	64414	6.41×10^{-5}	26050
39	0.0000000e+00	1.79×10^{-8}	2938	3.42×10^{-5}	134562
40	-1.0153200e+01	-10.15234	6618	-10.15234	5866
41	-1.0402940e+01	-10.40201	2298	-10.40201	2604
42	-1.0536410e+01	-10.53544	2498	-10.53544	3324
43	-1.8673091e+02	-186.72944	806	-186.72944	1684
44	0.0000000e+00	1.15×10^{-5}	112	1.15×10^{-5}	190
45	0.0000000e+00	2.87×10^{-5}	392	2.87×10^{-5}	1400
46	0.0000000e+00	5.74×10^{-5}	1054	5.74×10^{-5}	27566
47	0.0000000e+00	7.95×10^{-6}	274	7.95×10^{-6}	318
48	0.0000000e+00	3.73×10^{-5}	1678	3.73×10^{-5}	2218
49	0.0000000e+00	9.11×10^{-6}	3636	9.11×10^{-6}	9868
50	-5.0000000e+01	-49.99218	1324	-49.99218	1942
51	-2.1000000e+02	-209.98223	168432	-209.96002	279324
52	0.0000000e+00	0.00000	78	0.00000	78
53	0.0000000e+00	0.00000	22498	0.00000	24150
54	0.0000000e+00	9.13966	1284	9.13966	1410
Average		34062.037			38966.519
Median		1037.000			1575.000

and sometimes does not reach a comparable minimum function value to that obtained in BIRECT for certain test problems. This seems inappropriate and makes the comparison not favorable to our results. This is the case, for example, of *Ackley* test problems (No. 1–3).

At the same time, it requires fewer function evaluations than in DIRECT and DIRECT-1 algorithms. Also, the median value is smallest using BIRECT-V1 (921.000), compared to BIRECT-V (1681.000), DIRECT-1 (1752) and DIRECT (3810) algorithms.

On the other hand, our framework gives better results on the basis of the best (minimum) function value for almost all instances compared to both versions of BIRECT. In general, the overall average number of objective functions obtained with BIRECT-V algorithm is approximately 61, 11% (33 out of 54). To confirm the above-mentioned fact, it can be seen from Table 3, that the situation changes completely when the best-found objective function value $f(\bar{x})$ found by the BIRECT algorithm is used as a known optimal (minimal) value (f^*). Both BIRECT-V and BIRECT-V1 algorithms give on average significantly better results compared to the original BIRECT- and BIRECT-1 algorithms.

The same is observed, especially for some problems (for $n = 10$ cases), as for *Michalewics* (No.26), and *Zakharov* (No.54) test problems, while others have reached exactly the known optimal (minimal) value (f^*). This is the case of the following test problems: *Perm* (No.27), *Power Sum* (No.30), *Rastrigin* (No.31–

33), and *Zakharov* (No.52, 53). These results are confirmed by comparing the value of the global minimizer \mathbf{x}_{\min} from the libraries ([9], [53], [48]), and the value of \bar{x} generated by the algorithm (see Table 4).

More precisely, for the case of the problems: *Michalewics* (No.26), we found $x(10) = [1.57079632679490]$, *Perm* (No.27), the global minimizer found is $\mathbf{x}_{\min} = [1,2,3,4]$, *Power Sum* (No.30), the global minimum is 0, which is attained at $[2,1,3,2]$, *Rastrigin* (No.32), and *Zakharov* (No.52-53) test problems, the global minimum is 0, which is attained at $\mathbf{x}_{\min} = 0$. This situation arises occasionally, where at the early stages of the sampling process, the algorithm samples near a global optimum. Moreover, for some test problems, e.g., (*Dixon and Price* (No.13), *Michalewics* (No.25), *Powell* (No.29), *Schewefel* problem (No.39), *Trid* (No.51), as previously pointed out, we observed an excessive number of function evaluations. In this case, we observe the following situations:

- There is no improvement in the best function value after many consecutive iterations. The algorithm suffers from getting close to a global minimizer, and the objective function seems to be stagnating around a certain value, which may be a local optimum.
- An increasing number of evaluations (per iteration) is observed during the iteration's progress, as shown, e.g., in Fig. 4.

Notice that these situations are typical for diagonal-

Algorithm 1 The BIRECT-V algorithm

```

1: BIRECT-V ( $f, D, opt$ );
   Input: Objective function:  $f$ , search-space:
    $D$ , tolerance:  $\epsilon_{pe}$ , the maximal number
   of function evaluations:  $M_{max}$ , and the maxi-
   mal number of iterations:  $K_{max}$ ;
   Output: Global minimum:  $f_{min}$ , global min-
   imizer:  $x_{min}$ , and performance measures:
    $m, k$  and  $pe$  (if needed);


---


2: Normalize the search space  $D$  to be the unit hyper-
   cube  $\bar{D}$ ;
3: Initialize  $\mathbf{t}^1 = (1/3, \dots, 1/3)^T$  and  $\mathbf{v}^1 =$ 
    $(1, \dots, 1)^T$ ,  $m = 1, k = 1, \mathbb{I}_k = \{1\}$  and  $pe$ ;
    $\triangleright pe$  defined in Eq. (9)
4: Evaluate  $f(\mathbf{t}^1)$  and  $f(\mathbf{v}^1)$ , and set  $f_{min} =$ 
    $\min \{f(\mathbf{t}^1), f(\mathbf{v}^1)\}$ ,  $x_{min} = \operatorname{argmin}_{x \in \{\mathbf{t}^i, \mathbf{v}^i\}} f(x)$ ;
5:
6: while  $pe > \epsilon_{pe}, m < M_{max}, k < K_{max}$  do
7:   Identify the index set  $\mathbb{P}_k \subseteq \mathbb{I}_k$  of potentially op-
   timal hyper-rectangles (POHs) applying Inequa-
   tions (Ineq. (3); Ineq. (4));
8:   Select at most one POH from each group
   ; // Only in BIRECT-V1
9:   Set  $\mathbb{I}_k = \mathbb{I}_k \setminus \{\mathbb{P}_k\}$ ;
10:
11:   for  $i \in \mathbb{P}_k$  do
12:     Select the branching variable br (coordinate
   index) using Eq. (6);
13:     Divide  $\bar{D}^i$  into a two new hyper-rectangles
    $\bar{D}^{m+1}$  and  $\bar{D}^{m+2}$ ;
14:     Create the new sampling points  $\mathbf{t}^{m+1}$  and
    $\mathbf{v}^{m+2}$ ;  $\triangleright$  see illustration. ??;
15:     Evaluate  $f(\mathbf{t}^{m+1})$  and  $f(\mathbf{v}^{m+2})$ 
16:     Set  $f_{min}^{m+1} = \min \{f(\mathbf{t}^{m+1}), f(\mathbf{v}^{m+1})\}$  and
    $f_{min}^{m+2} = \min \{f(\mathbf{t}^{m+2}), f(\mathbf{v}^{m+2})\}$ ;
17:     Update the partition set  $\mathbb{I}_k = \mathbb{I}_k \cup \{m +$ 
    $1, m + 2\}$ ;
18:
19:     if  $f_{min}^{m+1} \leq f_{min}$  or  $f_{min}^{m+2} \leq f_{min}$  then
20:       Update  $f_{min}$  and  $x_{min}$ ;
21:       Update performance measures:  $k, m$  and  $pe$ ;
22: Return :  $f_{min}, x_{min}$  and algorithmic performance
   measures:  $m, k$  and  $pe$ .

```

based algorithms as well as for DIRECT-type algorithms. A detailed review could be found in [13]. Let us illustrate the above situations in the case of our sampling strategy. Assume that a global minimum is near one of the two sampled points located at $1/3$ and $2/3$ along one of the diagonals of a hyper-rectangle. This situation is in favor of BIRECT, since it samples one of these two points per hyper-rectangle. However, for the BIRECT-V algorithm, it may produce many unnecessary sampling points of the objective function at vertices before this optimum is reached. Every vertex could be shared up to 2^n hyper-rectangles, where the function

has been re-evaluated. In this case, the algorithm takes significantly longer than usual to find a good solution close to the global optimum. This can be observed from the results given in Table 5, where the two algorithms reached approximately the same best function value in some situations.

In the opposite scenario, i.e., if the global optimum point is located at the vertex of a hyper-rectangle, BIRECT- has a contrary impact to the previous situation. As the optimization proceeds, BIRECT-V requires fewer function evaluations than BIRECT, since many adjacent hyper-rectangles could share the same vertex.

In contrast to the previous situations, the same objective function value can be attained in many different points of the feasible domain, as in the case of the *Branin* test problem (No.9), where $\mathbf{x}_{min} = [3.13965, 2.275]$ for BIRECT-V, while for BIRECT, $\mathbf{x}_{min} = [9.42383, 2.471]$. This situation is current for multimodal problems (having multiple global minima), symmetrical problems, and (convex) quadratic test problems. Therefore, BIRECT-V requires fewer function evaluations, thus leading to a much larger set of selected potentially optimal hyper-rectangles having the same size and objective function value.

For the problems where BIRECT-V failed to converge most often, we suggested a modification to the original optimization domain to obtain a good approximation reasonably closer to the real (known) global optimum. The performance of the BIRECT-V algorithm is better compared to the original results. It is clear that this strategy does not overcome the situation in a proper way, but it allows the algorithm to avoid unnecessary sampling of objective function points at vertices and reduces considerably the number of function evaluations.

It should be stressed that we did not adopt any specific rule or known method for how the optimization domain is modified. Just slightly modify the domain until we find a minimizer close to the known solution, or at least to the one obtained by BIRECT. For example, For the *Schewefel* problem (No. 39), we obtained $\mathbf{x}_{min} = [420.9635416667]$ for BIRECT-V, and $\mathbf{x}_{min} = [420.9686279297]$ for BIRECT-V1. The domain was modified up to $[-500, 700]^{10}$, see [50, 42, 43].

Note that some results reported in Table 5, and Table 4 could be improved more and more, e.g., *Ackley* problem 1, 2 and 3 could be improved to get $f(\bar{x}) = 1.27161957e - 05$, with a global minimizer: $\mathbf{x}_{min} = [0.0000031789, \dots]$. Also, it is shown that some problems are sensitive to the domain modification, while others don't really require such a modification.

From Table 5, the numerical results prove that both BIRECT-V1 and BIRECT-V algorithms produce the best results based on the best found objective function value, with about 89% (48 out of 54) for BIRECT-V1, and 87% (47 out of 54) for BIRECT-V. On the other hand, we observe that the number of function evaluations is most often smallest for the BIRECT (for about 33 out of 54 of the test problems) and (30 out of 54) of

the BIRECT-1 problems) when compared to BIRECT-V and BIRECT-V1 respectively, in particular for the test problems having the same minimum value.

To conclude this comparison, it is important to notice that, despite the excessive number of evaluations due to many unnecessary sampling points at some shared vertices, BIRECT-V1 produces the best results in terms of the lowest function values and, on average, the almost smallest number of function evaluations compared to other algorithms.

4 Conclusions and Future Works

This paper proposes a new diagonal partitioning strategy for global optimization problems. A modification of the BIRECT algorithm based on bisection and a novel sampling scheme, contrary to the most DIRECT-type algorithms, where the evaluation of the objective function at vertices of hyper-rectangles are not suitable for bisection. The newly introduced BIRECT-V and its variant BIRECT-V1 were compared against BIRECT, BIRECT-1, and two DIRECT-type algorithms[27, 28].

The experimental results revealed that the new sampling scheme gives significantly better results for almost all test problems, particularly when the feasible domain is modified.

Further considerations may be investigated using additional assumptions to improve this version. One of these possible improvements is to evaluate the objective function only once at each vertex of each hyper-rectangle, where the objective function values at vertices could be stored in a special vertex database, thus avoiding re-evaluation of the objective function at certain shared vertices in adjacent hyper-rectangles. Another feature, as shown during the previous test process, is to find a specific rule about how the change in the original optimization domain should be applied in order to improve the performance of the BIRECT-V algorithm (see [48, 46, 45, 54]).

Finally, the results could also be extended to other test problems from [42]. All these observations may be considered for future research directions.

Acknowledgement: We would like to express our gratitude to Pr. M. Bentobache from who provided various suggestions for improvement of the results reported in Table 2.

References

- [1] CUSTÓDIO, A. L., ROCHA, H., AND VICENTE, L. N. Incorporating minimum frobenius norm models in direct search. *Computational Optimization and Applications* 46, 2 (2010), 265–278.
- [2] DANILIN, Y. M., AND PIYAVSKII, S. An algorithm for finding the absolute minimum. *Theory of Optimal Decisions* 2 (1967), 25–37.
- [3] DI SERAFINO, D., LIUZZI, G., PICCIALLI, V., RICCIO, F., AND TORALDO, G. A modified dividing rectangles algorithm for a problem in astrophysics. *Journal of optimization theory and applications* 151 (2011), 175–190.
- [4] FINKEL, D. E. *Global optimization with the DIRECT algorithm*. North Carolina State University, 2005.
- [5] FINKEL, D. E., AND KELLEY, C. Additive scaling and the direct algorithm. *Journal of Global Optimization* 36 (2006), 597–608.
- [6] FLOUDAS, C. A. *Deterministic global optimization: theory, methods and applications*, vol. 37. Springer Science & Business Media, 2013.
- [7] GABLONSKY, J. M., AND KELLEY, C. T. A locally-biased form of the direct algorithm. *Journal of Global optimization* 21 (2001), 27–37.
- [8] GABLONSKY, J. M. X. *Modifications of the DIRECT algorithm*. North Carolina state university, 2001.
- [9] HEDAR, A. Test functions for unconstrained global optimization, 2005.
- [10] HORST, R., PARDALOS, P. M., AND VAN THOAI, N. *Introduction to global optimization*. Springer Science & Business Media, 2000.
- [11] HORST, R., AND TUY, H. *Global optimization: Deterministic approaches*. Springer Science & Business Media, 2013.
- [12] JONES, D. R. Direct global optimization algorithm. *Encyclopedia of optimization* 1, 1 (2009), 431–440.
- [13] JONES, D. R., AND MARTINS, J. R. The direct algorithm: 25 years later. *Journal of Global Optimization* 79, 3 (2021), 521–566.
- [14] JONES, D. R., PERTTUNEN, C. D., AND STUCKMAN, B. E. Lipschitzian optimization without the lipschitz constant. *Journal of optimization Theory and Applications* 79 (1993), 157–181.
- [15] KUDELA, J. A critical problem in benchmarking and analysis of evolutionary computation methods. *Nature Machine Intelligence* 4, 12 (2022), 1238–1245.
- [16] KUDELA, J. Benchmarking state-of-the-art direct-type methods on the bbob noiseless testbed. In *GECCO '23 Companion: Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023), pp. 1620–1627.
- [17] KUDELA, J., AND JURICEK, M. Computational and exploratory landscape analysis of the gkls generator. In *GECCO '23 Companion: Proceedings of the Companion Conference on Genetic and Evolutionary Computation* (2023), pp. 443–446.
- [18] KVASOV, D. E., AND SERGEYEV, Y. D. Lipschitz gradients for global optimization in a one-point-based partitioning scheme. *Journal of Computational and Applied Mathematics* 236, 16 (2012), 4042–4054.
- [19] LIBERTI, L., AND KUCHERENKO, S. Comparison of deterministic and stochastic approaches to global optimization. *International Transactions in Operational Research* 12, 3 (2005), 263–285.

Table 4: Global minimizer found by the BIRECT-V algorithm using Hedjar test problems [9] with modified domain from Table 5.

Problem number (from Table 1)	Dimension n	Modified domain \bar{D}	Global minimizer found by BIRECT-V	Globally optimal known solution (Source [9, 53, 42])
1, 2, 3	2, 5, 10	$[-1.5, 32]^n$	$[0.000038147, 3.000000000, 0.4980468750]$	$[0]$ $[3; 0.5]$
4	2	$[-100, 110.7]^2$	$[0.0001953125, 0.9987304687, 3.0008789062]$	$[0; 0]$
5, 6, 7	2	$[-10, 10, 1]^2$	$[3.1396484375, 2.2753906250]$	$[1; 3]$
8	2	$[-10, 10, 1]^2$	$[0.9993489583, 1.0006510417]$	$[1; 1; 1; 1]$
9	2	$[-10, 10, 1]^2$	$[0.9993489583, 1.0006510417]$	$[1; 1; 1; 1]$
10	4	$[-10, 10, 1]^4$	$[0.9993489583, 1.0006510417]$	$[1; 1; 1; 1]$
11	2	$[-10, 10, 4.554]^2$	$[1.0033203125, -0.7069335937]$	$[2(-((2^i-2)/(2^i)))]$
12	5	$[-10, 40, 12, 30, 1]^5$	$[1.006, 0.709, 0.594, 0.545, 0.523]$	$[2(-((2^i-2)/(2^i)))]$
13	10	$[-10, 12]^{10}$	$[1.002, 0.708, 0.595, 0.544, 0.521, 0.510, 0.505, 0.502, 0.501, 0.501]$	$[2(-((2^i-2)/(2^i)))]$
14	2	$[-10, 10, 1]^2$	$[3.1412760417, 0.000000000, -1.000000000]$	$[\pi; \pi]$
15	2	$[-10, 10, 1]^2$	$[0.000000000, -1.000000000]$	$[0; -1]$
16	2	$[-10, 10, 1]^2$	$[0.0006357829, -0.0010172526]$	$[0]$
17	3	$[-10, 10, 1]^3$	$[0.114, 0.557, 0.854]$	$[0.115; 0.556; 0.852]$
18	6	$[-10, 10, 1]^6$	$[0.203, 0.148, 0.476, 0.273, 0.312, 0.656]$	$[0.202, 0.150, 0.477, 0.275, 0.312, 0.657]$
19	2	$[-10, 10, 1]^2$	$[-0.0911458333, 0.7096354167]$	$[-0.090; 0.713]$
20	2	$[-10, 10, 5]^2$	$[1.0027604167, 1.0027604167]$	$[1]$
21	5	$[-10, 10, 5]^5$	$[0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333]$	$[1]$
22	10	$[-10, 10, 5]^{10}$	$[0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333, 0.9973958333]$	$[1]$
23	2	$[-10, 10, 1]^2$	$[0.0260416667, 0.0260416667]$	$[0]$
24	2	$[-10, 10, 1]^2$	$[2.203, 1.571]$	$[2.203, 1.571]$
25	5	$[-1.04, \pi]^5$	$[2.203, 1.571, 1.285, 1.924, 1.720]$	$[2.203, 1.571, 1.285, 1.923, 1.720]$
26	10	$[-1.04, \pi]^{10}$	$[2.209, 1.571, 1.288, 1.117, 0.982, 1.571, 0.834, 2.356, 0.736, 1.571]$	$[2.203, 1.571, 1.285, 1.923, 1.720, 1.571, 1.454, 1.756, 1.656, 1.571]$
27	4	$[-10, 10, 1]^4$	$[1, 2, 3, 4]$	$[1, 2, 3, 4]$
28	4	$[-10, 10, 1]^4$	$[-0.021, 0.002, -0.039, -0.039]$	$[0]$
29	8	$[-4, -4, 0]^{18}$	$[0.009, -0.001, 0.005, -0.005, -0.050, 0.005, 0.005, 0.005]$	$[0]$
30	4	$[-10, 10, 1]^4$	$[1.001, 2.000, 2.000, 3.000]$	$[2.000, 1.000, 3.000, 2.000]$
31	2	$[-10, 10, 1]^2$	$[-0.0003483073, 0.0001139323, 0.0001000000, 0.0001000000]$	$[0]$
32	5	$[-5, 12, 5, 30]^5$	$[0.0001139323, 0.0001139323, 0.0001000000, 0.0001000000, 0.0001000000]$	$[0]$
33	10	$[-5, 12, 5, 12]^{10}$	$[0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000, 0.0001000000]$	$[0]$
34	2	$[-10, 10, 1]^2$	$[1.0009765625, 0.9997558594, 0.9998168945, 0.9998168945]$	$[1]$
35	5	$[-10, 10, 1]^5$	$[0.9997558594, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945]$	$[1]$
36	10	$[-10, 10, 1]^{10}$	$[0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945, 0.9998168945]$	$[1]$
37, 38	2, 5	$[-519, 519]^n$	$[420.96874737558, 420.96874737558]$	$[420.96874737558, 420.96874737558]$
39	10	$[-500, 650]^{10}$	$[420.96874737558, 420.96874737558]$	$[420.96874737558, 420.96874737558]$
40	4	$[-519, 519]^4$	$[4.001, 4.001, 4.001, 3.997]$	$[4.000; 4.000; 4.000; 4.000]$
41	4	$[-519, 519]^4$	$[4.001, 4.001, 3.997, 3.997]$	$[4.000; 4.001; 3.999; 3.999]$
42	4	$[-519, 519]^4$	$[4.001, 4.001, 3.997, 3.997]$	$[4.001; 4.000; 3.999; 3.999]$
43	2	$[-5, 12, 5, 12]^2$	$[-1.426, -0.801]$	$[4.858, -7.083]$
44	2	$[-5, 12, 5, 12]^2$	$[0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333]$	$[0]$
45	5	$[-5, 12, 5, 12]^5$	$[0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333]$	$[0]$
46	10	$[-5, 12, 5, 12]^{10}$	$[0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333, 0.0023958333]$	$[0]$
47	2	$[-10, 11, 5]^2$	$[0.0016276042, -0.0065104167]$	$[0]$
48	5	$[-10, 10, 5]^5$	$[0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042]$	$[0]$
49	10	$[-10, 10, 5]^{10}$	$[0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042, 0.0016276042]$	$[0]$
50	6	$[-36.5, 36.5]^6$	$[5.9880, 9.980, 11.976, 11.976, 9.980, 5.988]$	$[i * (6 + 1 - i)]$
51	10	$[-120, 120]^{10}$	$[10.000, 17.969, 23.984, 27.969, 29.922, 29.922, 27.969, 24.062, 17.969, 10.000]$	$[i * (10 + 1 - i)]$
52	2	$[-5, 12]^2$	$[0.0026041667, 0.0026041667]$	$[0]$
53	5	$[-5, 10, 0.1774]^5$	$[0.0010247461, 0.0010247461, 0.0010247461, 0.0010247461, 0.0010247461]$	$[0]$
54	10	$[-5, 13]^{10}$	$[0.000, 0.125, 0.000, 0.250, 0.250, -1.000, 0.000, 0.129, 0.129, 0.125]$	$[0]$

[20] LIU, H., XU, S., WANG, X., WU, J., AND SONG, Y. A global optimization algorithm for simulation-based problems via the extended direct

scheme. *Engineering Optimization* 47, 11 (2015), 1441–1458.

[21] LIU, Q., AND CHENG, W. A modified direct al-

Table 5: Comparison between BIRECT-V1, BIRECT-V, BIRECT-1, BIRECT, DIRECT-1, and DIRECT algorithms.

Problem No.	BIRECT-V1		BIRECT-V		BIRECT-1		BIRECT		DIRECT-1		DIRECT	
	$f(\#)$	f.eval.	$f(\#)$	f.eval.	$f(\#)$	f.eval.	$f(\#)$	f.eval.	$f(\#)$	f.eval.	$f(\#)$	f.eval.
1	1.52×10^{-5}	218	1.52×10^{-5}	260	2.54×10^{-5}	176	2.54×10^{-5}	202	7.53×10^{-5}	185	7.53×10^{-5}	255
2	1.52×10^{-5}	524	1.52×10^{-5}	2728	2.54×10^{-5}	454	2.54×10^{-5}	1268	7.53×10^{-5}	1777	7.53×10^{-5}	8845
3	1.52×10^{-5}	1280	137040	137040	2.54×10^{-5}	874	2.54×10^{-5}	47792	7.53×10^{-5}	> 500000	7.53×10^{-5}	80927
4	8.77×10^{-5}	640	8.77×10^{-5}	1034	9.17×10^{-5}	436	9.17×10^{-5}	436	9.29×10^{-5}	247	9.29×10^{-5}	655
5	1.83×10^{-6}	254	3.17×10^{-5}	524	4.02×10^{-5}	468	4.02×10^{-5}	476	3.09×10^{-5}	205	3.09×10^{-5}	327
6	1.53×10^{-6}	252	1.53×10^{-6}	284	3.35×10^{-5}	472	3.35×10^{-5}	478	2.58×10^{-5}	233	2.58×10^{-5}	345
7	2.88×10^{-6}	284	2.88×10^{-6}	282	3.67×10^{-5}	474	3.67×10^{-5}	480	8.21×10^{-5}	573	8.21×10^{-5}	693
8	2.99×10^{-6}	300	2.99×10^{-6}	334	6.10×10^{-5}	188	6.10×10^{-5}	194	6.58×10^{-5}	215	6.58×10^{-5}	295
9	0.39791	656	0.39791	492	0.39790	242	0.39790	242	0.39789	169	0.39789	195
10	9.82×10^{-5}	2320	9.82×10^{-5}	1910	9.82×10^{-5}	794	9.82×10^{-5}	794	3.83×10^{-5}	3379	6.08×10^{-5}	6585
11	4.01×10^{-5}	810	4.01×10^{-5}	784	4.84×10^{-5}	722	4.84×10^{-5}	722	5.32×10^{-5}	485	6.25×10^{-5}	513
12	7.57×10^{-5}	10872	7.57×10^{-5}	8446	7.15×10^{-5}	4060	7.15×10^{-5}	4060	6.45×10^{-5}	54843	6.45×10^{-5}	19661
13	7.02×10^{-5}	85492	7.60×10^{-5}	50922	9.52×10^{-5}	162862	9.52×10^{-5}	164826	0.66667	> 500000	5.79×10^{-5}	372619
14	-0.99999	180	-0.99999	1082	-0.99999	480	-0.99999	16420	-0.99999	6851	-0.99999	32845
15	3.00000	28	3.00000	28	3.00019	274	3.00019	274	3.00009	115	3.00009	191
16	4.61×10^{-7}	8456	4.61×10^{-7}	9162	7.76×10^{-7}	5106	7.76×10^{-7}	5106	4.84×10^{-6}	8379	4.84×10^{-6}	9215
17	-3.86245	200	-3.86245	208	-3.86242	352	-3.86242	352	-3.86245	111	-3.86245	199
18	-3.32214	542	-3.32214	542	-3.32206	764	-3.32206	764	-3.32207	295	-3.32207	371
19	-1.03154	202	-1.03154	334	-1.03154	190	-1.03154	334	-1.03162	137	-1.03162	321
20	9.03×10^{-6}	136	9.03×10^{-6}	154	9.09×10^{-5}	152	9.09×10^{-5}	152	2.10×10^{-5}	77	2.10×10^{-5}	105
21	1.83×10^{-5}	454	1.83×10^{-5}	558	1.83×10^{-5}	660	1.83×10^{-5}	1024	3.65×10^{-5}	359	3.65×10^{-5}	705
22	3.54×10^{-5}	1182	3.54×10^{-5}	7440	3.55×10^{-5}	1698	3.55×10^{-5}	7904	9.62×10^{-5}	5297	6.23×10^{-5}	5589
23	2.71×10^{-5}	148	2.71×10^{-5}	208	2.71×10^{-5}	90	2.71×10^{-5}	94	3.81×10^{-5}	71	3.81×10^{-5}	107
24	-1.80130	184	-1.80130	314	-1.80118	126	-1.80118	126	-1.80127	45	-1.80127	69
25	-4.68744	8430	-4.68744	7472	-4.68736	101942	-4.68736	73866	-4.68721	26341	-4.68721	13537
26	-8.60559	> 500000	-7.55575	> 500000	-7.32661	> 500000	-7.32661	> 500000	-7.84588	> 500000	-7.87910	> 500000
27	0.00132	> 500000	0.00189	> 500000	0.00203	> 500000	0.00203	> 500000	0.04054	> 500000	0.04355	> 500000
28	4.59×10^{-5}	2786	4.59×10^{-5}	1678	4.86×10^{-5}	1832	4.86×10^{-5}	2114	6.52×10^{-5}	32331	9.02×10^{-5}	14209
29	9.00×10^{-5}	2872	9.00×10^{-5}	3072	9.71×10^{-5}	92884	9.71×10^{-5}	99514	0.02488	> 500000	0.02142	> 500000
30	0.00000	204	0.00000	40788	9.00×10^{-5}	1718	9.00×10^{-5}	10856	0.03524	> 500000	0.00215	> 500000
31	4.81×10^{-5}	774	4.81×10^{-5}	958	4.81×10^{-5}	154	4.81×10^{-5}	180	2.30×10^{-5}	1727	2.30×10^{-5}	987
32	1.29×10^{-5}	9126	1.29×10^{-5}	11008	1.18×10^{-5}	472	1.18×10^{-5}	1394	4.97479	> 500000	4.97479	> 500000
33	1.98×10^{-5}	124	1.98×10^{-5}	1454	2.36×10^{-5}	1250	2.36×10^{-5}	40254	4.97479	> 500000	4.97479	> 500000
34	9.65×10^{-5}	698	9.65×10^{-5}	718	9.65×10^{-5}	242	9.65×10^{-5}	242	9.65×10^{-5}	285	9.65×10^{-5}	1621
35	2.41×10^{-5}	2444	2.41×10^{-5}	2972	2.41×10^{-5}	1494	2.41×10^{-5}	1700	5.75×10^{-5}	2703	8.80×10^{-5}	20025
36	3.05×10^{-5}	19134	3.05×10^{-5}	31430	5.42×10^{-5}	4590	5.42×10^{-5}	10910	8.29×10^{-5}	74071	8.29×10^{-5}	174529
37	1.37×10^{-7}	492	1.37×10^{-7}	564	5.64×10^{-5}	210	5.64×10^{-5}	236	2.88×10^{-5}	341	2.88×10^{-5}	255
38	3.42×10^{-7}	24272	3.42×10^{-7}	16704	6.41×10^{-5}	1422	6.41×10^{-5}	7210	7.21×10^{-5}	322039	7.21×10^{-5}	31999
39	1.77×10^{-8}	1492	1.77×10^{-8}	86306	1.30×10^{-6}	58058	1.30×10^{-6}	315960	1269.34444	> 500000	1187.63199	> 500000
40	-10.15234	6618	-10.15234	5866	-10.15234	1200	-10.15234	147	-10.152234	155	-10.152234	145
41	-10.40201	2298	-10.40201	2604	-10.40269	1224	-10.40269	1180	-10.40196	141	-10.40196	145
42	-10.53544	2498	-10.53544	3324	-10.53618	1158	-10.53618	1140	-10.53539	189	-10.53539	2967
43	-186.72944	806	-186.72944	1684	-186.72441	2114	-186.72441	1780	-186.72153	2043	-186.72153	2967
44	1.15×10^{-5}	112	1.15×10^{-5}	190	1.15×10^{-5}	108	1.15×10^{-5}	118	8.74×10^{-5}	91	8.74×10^{-5}	209
45	2.87×10^{-5}	392	2.87×10^{-5}	1400	2.87×10^{-5}	288	2.87×10^{-5}	712	7.49×10^{-5}	465	7.49×10^{-5}	4653
46	5.74×10^{-5}	1054	5.74×10^{-5}	27566	5.74×10^{-5}	784	5.74×10^{-5}	16974	9.63×10^{-5}	2057	9.63×10^{-5}	99123
47	8.74×10^{-5}	248	8.74×10^{-5}	280	7.94×10^{-6}	226	7.94×10^{-6}	244	3.52×10^{-5}	77	3.52×10^{-5}	107
48	3.97×10^{-5}	1354	3.97×10^{-5}	1776	3.97×10^{-5}	836	3.97×10^{-5}	1034	7.19×10^{-5}	411	7.19×10^{-5}	833
49	9.35×10^{-5}	3394	9.35×10^{-5}	9244	9.11×10^{-6}	3366	9.11×10^{-6}	7688	7.76×10^{-5}	1809	7.76×10^{-5}	8133
50	-49.99788	1312	-49.99788	1662	-49.99512	1138	-49.99512	1506	-49.99525	8731	-49.99525	5693
51	-209.98779	3114	-209.98779	11878	-209.98007	24716	-209.98007	30100	-209.92644	> 500000	-209.98085	90375
52	2.88×10^{-5}	156	2.88×10^{-5}	162	2.88×10^{-5}	338	2.88×10^{-5}	502	7.95×10^{-5}	209	7.95×10^{-5}	237
53	6.43×10^{-5}	3810	6.43×10^{-5}	4060	6.44×10^{-5}	27364	6.44×10^{-5}	20974	0.11921	> 500000	9.71×10^{-5}	316827
54	2.607286	> 500000	2.607286	> 500000	9.41133	> 500000	9.41133	> 500000	16.47703	> 500000	28.96394	> 500000
Average		30844.296		37072.0371		37283.85		44520.52		121484.19		98677.70
Median		808.00		1681.00		789.00		1190.00		1752.00		3810.00

gorithm with bilevel partition. *Journal of Global Optimization* 60, 3 (2014), 483–499. [22] LIU, Q., ZENG, J., AND YANG, G. Mrdirect: a multilevel robust direct algorithm for global opti-

- mization problems. *Journal of Global Optimization* 62, 2 (2015), 205–227.
- [23] LIUZZI, G., LUCIDI, S., AND PICCIALI, V. A direct-based approach exploiting local minimizations for the solution of large-scale global optimization problems. *Computational Optimization and Applications* 45 (2010), 353–375.
- [24] LIUZZI, G., LUCIDI, S., AND PICCIALI, V. A partition-based global optimization algorithm. *Journal of Global Optimization* 48 (2010), 113–128.
- [25] LIUZZI, G., LUCIDI, S., AND PICCIALI, V. Exploiting derivative-free local searches in direct-type algorithms for global optimization. *Computational Optimization and Applications* 65 (2016), 449–475.
- [26] MA, K., RIOS, L. M., BHOSEKAR, A., SAHINIDIS, N. V., AND RAJAGOPALAN, S. Branch-and-model: a derivative-free global optimization algorithm. *Computational Optimization and Applications* 85, 2 (2023), 337–367.
- [27] PAULAVIČIUS, R., CHITER, L., AND ŽILINSKAS, J. Global optimization based on bisection of rectangles, function values at diagonals, and a set of lipschitz constants. *Journal of Global Optimization* 71 (2018), 5–20.
- [28] PAULAVIČIUS, R., SERGEYEV, Y. D., KVASOV, D. E., AND ŽILINSKAS, J. Globally-biased direct algorithm with local accelerators for expensive global optimization. *Expert Systems with Applications* 144 (2020), 113052.
- [29] PAULAVIČIUS, R., AND ŽILINSKAS, J. Analysis of different norms and corresponding lipschitz constants for global optimization. *Technological and Economic Development of Economy* 12, 4 (2006), 301–306.
- [30] PAULAVIČIUS, R., AND ŽILINSKAS, J. *Simplicial global optimization*, vol. 1. Springer, 2014.
- [31] PAULAVIČIUS, R., ŽILINSKAS, J., AND GROTHEY, A. Parallel branch and bound for global optimization with combination of lipschitz bounds. *Optimization methods and software* 26, 3 (2011), 487–498.
- [32] PINTÉR, J. D. *Global optimization in action: continuous and Lipschitz optimization: algorithms, implementations and applications*, vol. 6. Springer Science & Business Media, 1995.
- [33] SERGEYEV, Y. D. On convergence of “divide the best” global optimization algorithms. *Optimization* 44, 3 (1998), 303–325.
- [34] SERGEYEV, Y. D. Efficient strategy for adaptive partition of n-dimensional intervals in the framework of diagonal algorithms. *Journal of Optimization Theory and Applications* 107 (2000), 145–168.
- [35] SERGEYEV, Y. D. Efficient partition of n-dimensional intervals in the framework of one-point-based algorithms. *Journal of optimization theory and applications* 124, 2 (2005), 503–510.
- [36] SERGEYEV, Y. D., AND KVASOV, D. Diagonal global optimization methods. fiz-matlit, moscow, 2008. *Russian. Zbl1282 90138*.
- [37] SERGEYEV, Y. D., AND KVASOV, D. E. Global search based on efficient diagonal partitions and a set of lipschitz constants. *SIAM Journal on Optimization* 16, 3 (2006), 910–937.
- [38] SERGEYEV, Y. D., AND KVASOV, D. E. Lipschitz global optimization. In *Cochran, J.J., Cox, L.A., Keskinocak, P., Kharoufeh, J.P., Smith, J.C. (eds.) Wiley Encyclopedia of Operations Research and Management Science (in 8 Volumes)* (2011), vol. 4, Wiley, pp. 2812–2828.
- [39] SERGEYEV, Y. D., AND KVASOV, D. E. On deterministic diagonal methods for solving global optimization problems with lipschitz gradients. In *Optimization, Control, and Applications in the Information Age: In Honor of Panos M. Pardalos’s 60th Birthday* (2015), Springer, pp. 315–334.
- [40] SERGEYEV, Y. D., AND KVASOV, D. E. *Deterministic global optimization: An introduction to the diagonal approach*. Springer, 2017.
- [41] SHUBERT, B. O. A sequential method seeking the global maximum of a function. *SIAM Journal on Numerical Analysis* 9, 3 (1972), 379–388.
- [42] STRIPINIS, L., AND PAULAVIČIUS, R. DIRECT-GOLib - DIRECT Global Optimization test problems Library, v1.1. Zenodo (2022). <https://doi.org/10.5281/zenodo.6491951>.
- [43] STRIPINIS, L., AND PAULAVIČIUS, R. Directgo: A new direct-type matlab toolbox for derivative-free global optimization. *ACM Transactions on Mathematical Software* 48, 4 (2022), 1–46.
- [44] STRIPINIS, L., AND PAULAVIČIUS, R. An empirical study of various candidate selection and partitioning techniques in the direct framework. *Journal of Global Optimization* (2022), 1–31.
- [45] STRIPINIS, L., AND PAULAVIČIUS, R. Experimental study of excessive local refinement reduction techniques for global optimization direct-type algorithms. *Mathematics* 10, 20 (2022), 3760.
- [46] STRIPINIS, L., AND PAULAVIČIUS, R. An extensive numerical benchmark study of deterministic vs. stochastic derivative-free global optimization algorithms. *arXiv preprint arXiv:2209.05759* (2022).
- [47] STRIPINIS, L., AND PAULAVIČIUS, R. Gendirect: a generalized direct-type algorithmic framework for derivative-free global optimization. *arXiv preprint arXiv:2309.00835* (2023).
- [48] STRIPINIS, L., AND PAULAVIČIUS, R. Lipschitz-inspired halrect algorithm for derivative-free global optimization. *Journal of Global Optimization* (2023), 1–31.
- [49] STRIPINIS, L., AND PAULAVIČIUS, R. Novel algorithm for linearly constrained derivative free global optimization of lipschitz functions. *Mathematics* 11, 13 (2023), 2920.

-
- [50] STRIPINIS, L., PAULAVIČIUS, R., AND KUDELA, J. DIRECTGO: A new DIRECT-type MATLAB toolbox for derivative free global optimization. GitHub (2022). <https://github.com/blockchain-group/DIRECTGO>.
- [51] STRIPINIS, L., PAULAVIČIUS, R., AND ŽILINSKAS, J. Improved scheme for selection of potentially optimal hyper-rectangles in direct. *Optimization Letters* 12 (2018), 1699–1712.
- [52] STRIPINIS, L., PAULAVIČIUS, R., AND ŽILINSKAS, J. Penalty functions and two-step selection procedure based direct-type algorithm for constrained global optimization. *Structural and Multidisciplinary Optimization* 59 (2019), 2155–2175.
- [53] SURJANOVIC, S., AND BINGHAM, D. Virtual library of simulation experiments: test functions and datasets. *Simon Fraser University, Burnaby, BC, Canada, accessed May 13* (2013), 2015.
- [54] TSVETKOV, E., AND KRYMOV, R. Pure random search with virtual extension of feasible region. *Journal of Optimization Theory and Applications* 195, 2 (2022), 575–595.
- [55] TUY, H., HOANG, T., HOANG, T., MATHÉMATICIEN, V.-N., HOANG, T., AND MATHEMATICIAN, V. *Convex analysis and global optimization*. Springer, 1998.
- [56] ZHIGLJAVSKY, A., AND ZILINSKAS, A. *Stochastic global optimization*, vol. 9. Springer Science & Business Media, 2007.