A New DIRECT-Type Algorithm Based on Bisection of Rectangles and Diagonal Sampling with the Pareto Approach

Mira Mustika, Salmah*, and Indarsih

***Abstract*—A novel DIRECT-type algorithm is proposed to tackle box-constrained optimization problems. The algorithm incorporates bisection partitioning and diagonal sampling procedures, utilizing the Pareto approach to identify potential hyperrectangles. Furthermore, each hyperrectangle's size is determined based on the length of its longest side, in accordance with the infinity norm. This combination enhances convergence speed, facilitating the generation of a global optimal solution. The proposed algorithm's effectiveness is evaluated through numerical experiments, with detailed results demonstrating its efficacy.**

***Index Terms*—Global optimization, DIRECT-type algorithm, bisection, diagonal sampling, Pareto approach.**

I. Introduction

THIS study addresses the global optimization problems of a real-valued function adhering to the Lipschitz condition (even in cases where the Lipschitz constant is unknown), formulated as

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

The feasible region $D = [\mathbf{a},\mathbf{b}] \subseteq \mathbb{R}^n$ is a hyperrectangle. The function $f(\mathbf{x})$ may exhibit nonlinearity, nonconvexity, and nondifferentiability.

One interesting problem is that the objective function in Eq. (1) is treated as a black-box function. A black-box function lacks a known or exploitable analytical expression, yet its function values within the feasible area are available [1]. Determining the derivative of a black-box function is impossible. Therefore, optimization algorithms relying on derivative information find limited utility in such scenarios.

The DIRECT method is a well-known derivative-free optimization algorithm that can solve black-box problems [2]. Different from other derivative-free algorithms inspired by nature, such as Genetic Algorithm (GA) [3], Ant Colony Algorithm [4], and Simulated Annealing [5], the DIRECT method is motivated by Lipschitz optimization [6], notably the Piyavskii-Shubert algorithm [7], [8]. The Piyavskii–Shubert algorithm requires knowledge of the Lipschitz constant. However, obtaining this constant for many optimization problems can be challenging. Jones et al. [2] introduced a modification that eliminates the need for prior knowledge of the Lipschitz constant of the objective function.

The DIRECT method partitions the search space into smaller hyperrectangles. At each iteration, several points are sampled simultaneously, some for local search and some for global search. This balance between local and global search does not require tuning parameters. Another strength of the DIRECT method is that it does not require an initial starting point during the initialization stage, unlike global optimization algorithms such as the filled function method [9].

In addition to its simplicity, the DIRECT method excels in finding convergence points for global optimal solutions [10]. The DIRECT method is also effective and efficient in solving low-dimensional global optimization problems (2–6 variables) [11]. Rios and Sahinidis [12] compared the DIRECT method with other derivative-free global optimization algorithms, revealing the superior accuracy of the DIRECT method in obtaining global optimum solutions.

Although the DIRECT method is fast in finding global optimum basins, it occasionally exhibits slowness in refining solutions with high accuracy [13], [14], [15]. In the original DIRECT [2], the center point of each hyperrectangle serves as the sample point. However, slow convergence ensues if the hyperrectangle containing the global optimum solution yields an unfavorable objective function value at the center point. This occurs because it is necessary to select other hyperrectangles of the same or larger size with superior objective function values at their center points. To address this limitation, some authors [16], [17], [18] modified the sampling procedure, employing a diagonal sampling scheme. Numerical results demonstrate that this modification yields highly competitive and promising performance compared to the original DIRECT and its well-known modifications.

To expedite solution refinement, some authors modified the original DIRECT method. First, the size of the hyperrectangle is expressed as the length of the longest side, in accordance with the use of the infinity norm [19], [20]. The original DIRECT uses the Euclidean distance between the center and the vertex points. This modification results in fewer hyperrectangles, thereby reducing the global barrier to local refinement. Another modification pertains to the hyperrectangle partition scheme, favoring bisection over trisection due to the hyperrectangle's shape [21]. With a bisection partition, fewer variations in hyperrectangle sizes are formed compared with a trisection partition.

Manuscript received November 29, 2023; revised June 18, 2024.

This work was supported by Kementerian Pendidikan, Kebudayaan, Riset, dan Teknologi Indonesia through the Doctoral Dissertation Research (Penelitian Disertasi Doktor) program with Grant Number 3105/UN 1/DITLIT/Dit-Lit/PT.O 1.03/2023.

M.Mustika is a doctoral student at Mathematics Department, Universitas Gadjah Mada, Yogyakarta, 55281, Indonesia (e-mail:mira.mustika@mail.ugm.ac.id).

*Salmah is a professor at Mathematics Department, Universitas Gadjah Mada, Yogyakarta, 55281, Indonesia (corresponding author to provide phone: +62 888-6835-039; fax: 0274 513339; e-mail: syalmah@yahoo.com).

Indarsih is an Associate Professor at Mathematics Department, Universitas Gadjah Mada, Yogyakarta 55281, Indonesia (e-mail: indarsih@ugm.ac.id).

One potential area for enhancing the original DIRECT method lies in the lack of comprehensive global search, resulting in delayed discovery of the global minimum [10]. This limitation arises from the omission of certain non-dominated hyperrectangles in both objective function and size within the procedure of potentially optimal hyperrectangles (POHs) in the original DIRECT. This observation prompted Mockus [22] to introduce the Pareto approach for hyperrectangle selection, culminating in the development of the Pareto–Lipschitzian optimization (PLO) algorithm and its subsequent modifications [23], [24]. The Pareto approach, adopted in this paper, facilitates the selection and exploration of more hyperrectangles than the DIRECT method, proving advantageous in parallel computing. Notably, this approach operates without the need for additional parameters.

In this paper, we introduce a new DIRECT-type algorithm to solve global optimization problems. The algorithm combines key strengths from various approaches: the Pareto approach in hyperrectangle selection, bisection partition, diagonal sampling schemes, and the use of the infinity norm for hyperrectangle measurement. In each iteration, hyperrectangles meeting the Pareto optimal criteria are selected and subsequently partitioned into two smaller hyperrectangles. This combination results in fewer hyperrectangle groups, providing more comprehensive objective function information and reducing the global drag in obtaining the optimum solution. We named this algorithm the PLOBi algorithm.

The organization of this paper is as follows: Section 2 provides a detailed description of the PLOBi algorithm. Section 3 discusses the convergence of the algorithm. Section 4 presents numerical experiments using the Hedar test set function [25], comparing the results with two other DIRECT-type algorithms. Finally, Section 5 presents conclusions.

II. Description of the PLOBi Algorithm

In this section, a detailed account of the PLOBi algorithm is presented. The PLOBi algorithm combines bisection and diagonal sampling, as in [18], with the Pareto optimal approach for selecting hyperrectangles. In the initialization step, the initial search space $D = [\mathbf{a},\mathbf{b}] \subseteq R^n$ is normalized to a unit hypercube, $\bar{D} = [\mathbf{0},\mathbf{1}]$. Two distinct points on the hypercube D's main diagonal are then selected. Denoting the length of the main diagonal as L_d, the two selected points are located at $\frac{1}{3}L_d$ and $\frac{2}{3}L_d$ from the end point of the diagonal. For the n-dimensional problem, during the initial iteration, the two sampling points are point $\mathbf{l} = (l_1, \ldots, l_n) = \left(\frac{1}{3}, \ldots, \frac{1}{3}\right)$ and point $\mathbf{u} = (u_1, \ldots, u_n) = \left(\frac{2}{3}, \ldots, \frac{2}{3}\right)$.

The PLOBi algorithm encompasses three key procedures in each iteration: selecting hyperrectangles for further exploration, sampling, and partitioning procedures on each chosen hyperrectangle. In the original DIRECT [2], hyperrectangles are chosen for subsequent iterations based on each hyperrectangle's estimated lower bound for $f(\mathbf{x})$. In the PLO algorithm [22] and its derivative, the PLOR algorithm [23], [24], the selection of Pareto optimal hyperrectangles is driven by the lower bound function. The authors in [24] defined dominance hyperrectangle and Pareto optimal by treating the two terms in the lower bound function as two simultaneously optimized functions. In the PLO and PLOR algorithms, each hyperrectangle is evaluated only at one point, namely, the center point. In contrast, the PLOBi algorithm evaluates two points in each hyperrectangle. Consequently, an adjustment to the definition of hyperrectangle dominance is necessary for the PLOBi algorithm, as outlined in Definition 2.1.

Definition 2.1: Suppose that in the most recent iteration, the search space $\bar{D} = [\mathbf{0},\mathbf{1}] \subseteq \mathbb{R}^n$ has been partitioned into hyperrectangles $\bar{D}_i = [\mathbf{a}_i, \mathbf{b}_i] \subseteq \bar{D}$ where $i \in \mathbb{I}_k$ represents the index of the hyperrectangles formed after k iterations. The points $\mathbf{l}_i$ and $\mathbf{u}_i$ denote two different sample points on the main diagonal of the hyperrectangle $\bar{D}_i$, and δ_i denote the measure of the hyperrectangle $\bar{D}_i$. Hyperrectangle $\bar{D}_j$ dominates hyperrectangle $\bar{D}_i, i, j \in \mathbb{I}_k$ (denoted $\bar{D}_j \prec \bar{D}_i$) if

$$\delta_j \geq \delta_i \text{ and} \min\{f(\mathbf{l}_j), f(\mathbf{u}_j)\} < \min\{f(\mathbf{l}_i), f(\mathbf{u}_i)\} \quad (2)$$

or

$$\delta_j > \delta_i \text{ and} \min\{f(\mathbf{l}_j), f(\mathbf{u}_j)\} \leq \min\{f(\mathbf{l}_i), f(\mathbf{u}_i)\} \quad (3)$$

where

$$\delta_j = \max_{p=1,\ldots,n} d_j^p = \max_{p=1,\ldots,n} \mid b_j^p - a_j^p \mid . \quad (4)$$

From Definition 2.1, if hyperrectangle $\bar{D}_j$ does not dominate hyperrectangle $\bar{D}_i$ for $i, j \in \mathbb{I}_k$, then Eqs. (2) and (3) are not satisfied. If no hyperrectangle $\bar{D}_j$ dominates $\bar{D}_i$ for all $i, j \in \mathbb{I}_k$, then the hyper-rectangle $\bar{D}_i$ is regarded Pareto optimal. The formal definition of Pareto optimal hyperrectangle is given in Definition 2.2 [24].

Definition 2.2: The hyperrectangle $\bar{D}_i, i \in \mathbb{I}_k$ is Pareto optimal if there is no hyperrectangle $\bar{D}_j$ for all $j \in \mathbb{I}_k$ dominating $\bar{D}_i$.

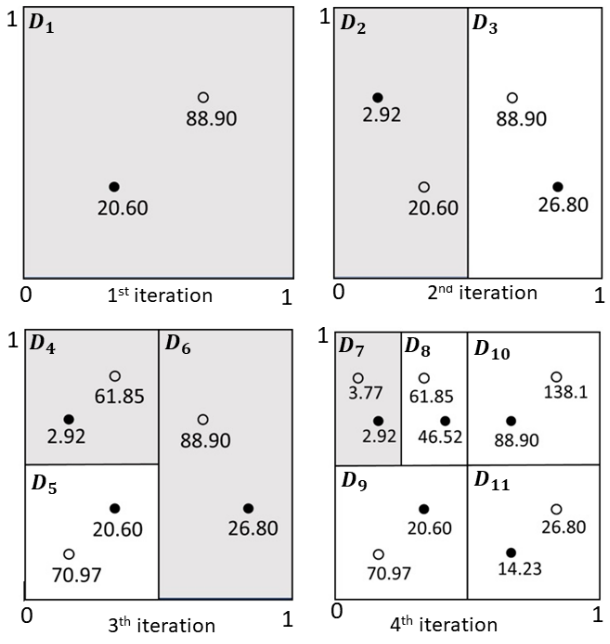

Figure 1. The first four iterations of partitioning and selecting of Pareto optimal hyperrectangles using the PLOBi algorithm for the Branin test problem.

The partition and diagonal sampling procedures in the PLOBi algorithm resemble those of the BIRECT algorithm [18]. The implementation of the PLOBi algorithm for the first four iterations of the 2D Branin function is illustrated in Fig. 1. The gray hyperrectangles represent Pareto optimal hyperrectangles. In the initialization stage, the objective function is evaluated at two points located on the main diagonal. Trivially, hypercube D_1 is chosen as the Pareto optimal hyperrectangle. Next, the hypercube is partitioned into two based on the smallest coordinate, namely the first coordinate (horizontal coordinate). Next, new observation points are sampled using the procedure described in [18].

Let l_i and u_i denote two points within the hyperrectangle D_i, where i epresents the hyperrectangle's index at an iteration. From Fig. 1, the black points within each hyperrectangle denote points with the best objective function values, i.e., those satisfying $\min\{f(l_i), f(u_i)\}$. In the 2^{nd} iteration in Fig. 1, D_2 and D_3 have the same size. Based on Eq. (2), D_2 dominates D_3 because $\min\{f(l_2), f(u_2)\} < \min\{f(l_3), f(u_3)\} \leftrightarrow 2.92 < 26.80$. Therefore, D_2 is selected as the optimal Pareto hyperrectangle. Analogously, in the 3^{th} iteration, D_4 dominates D_5, and none of the hyperrectangles dominate D_6. Thus, D_4 and D_6 are selected as the optimal Pareto. In the 4^{th} iteration, based on Eq. (4), all hyperrectangles have the same size. Therefore, based on Eq. (2), the hyperrectangle with the smallest objective function value, i.e., D_7 is selected as the optimal Pareto hyperrectangle.

The measure of the hyperrectangle j, δ_j in Eq. (4) corresponds to the infinity norm. Two hyperrectangles with different characteristics can belong to the same group using this measure. This allows for the formation of fewer groups of hyperrectangles with the same size, despite an increase in the number of hyperrectangles belonging to one group. In the fourth iteration (the fourth column in Fig. 1), characteristically, there are two types of hyperrectangles. However, according to Eq. (4), all the hyperrectangles formed possess the same maximum side length. In other words, there is only one hyperrectangle group and only one Pareto optimal hyperrectangle.

Fig. 2 presents a geometric interpretation of Definition 2.1 and Definition 2.2 of the PLOBi algorithm. The Fig. 3 illustrates the selection of POHs using the BIRECT algorithm for the Branin test function in the 10th iteration. Each point on both graphs represents a hyperrectangle. The horizontal axis denotes the size of the hyperrectangle (see Eq. (4)), and the vertical axis represents the best function value at the sample points for each hyperrectangle. The filled black circles represent the hyperrectangles selected for further searches. In the 10th iteration of the Branin test function, the group of hyperrectangles formed by the PLOBi algorithm is smaller than that of the BIRECT algorithm. This outcome translates to fewer rectangles being selected for subsequent searches, enabling a reduction in the number of functions evaluated.

III. Convergence of the PLOBi Algorithm

Many researchers [2], [17], [18] have discussed and proven the convergence properties of DIRECT-type algorithms. Generally, the PLOBi algorithm procedures follow those of the DIRECT and PLO algorithms. Therefore, the convergence of the PLOBi algorithm is determined by the convergence of the DIRECT-type algorithms.

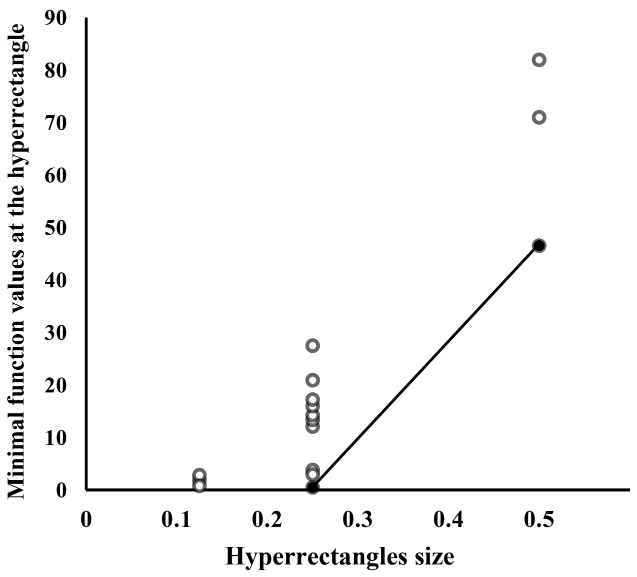

Figure 2. Geometric interpretation of Pareto optimal hyperrectangles selected by the PLOBi algorithm on the Branin test problem in the tenth iteration.

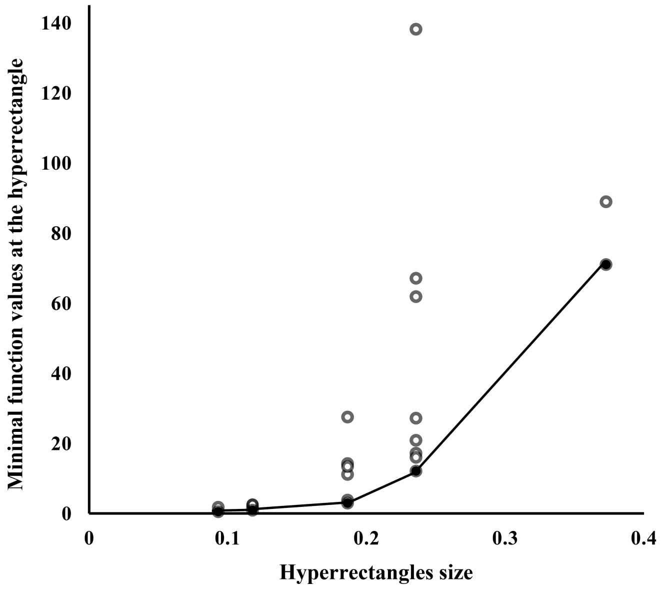

Figure 3. Geometric interpretation of potentially optimal hyperrectangles by the BIRECT algorithm on the Branin test problem in the tenth iteration.

In each iteration of the PLOBi algorithm, the hyperrectangle that satisfies the optimal Pareto is selected for further search. The following Lemma is provided to ensure that there is always a selected hyperrectangle in each iteration of the PLOBi algorithm.

Lemma 3.1: Let $\mathbb{I}_k$ represent the set of the hyperrectangle indices at iteration k. In each iteration, there is always the largest hyperrectangle that becomes Pareto optimal.

Proof: Let $\mathbb{I}_k^{\max} \subseteq \mathbb{I}_k$ represent the set of the hyperrectangle indices with the largest size, $\delta_{\max}^k$, at the iteration k. Based on Eqs. (2) and (3), there are no hyperrectangles $D_i, i \in \mathbb{I}_k$ where $\delta_i < \delta_{\max}^k$ which dominate hyperrectangle $D_{i_k^{\max}}, i_k^{\max} \in \mathbb{I}_k^{\max}$. Therefore, there always exists the largest hyperrectangle $D_{j_k^{\max}}$ that becomes Pareto optimal,

Algorithm 1 : The PLO-Bi algorithm

```
Input : $\epsilon_{\text{tol}}$ , $N_{f_{max}}$
Output : $f_{\min}, x_{\min}$, performance measure
The initial search space $D = [\mathbf{a}, \mathbf{b}]$ is normalized to the unit hypercube $\bar{D} = [\mathbf{0}, \mathbf{1}]$;
Initialization: $k = 1, \mathbb{I}_k = \{1\}, \mathbf{l}_1 = \left(\frac{1}{3}, \dots, \frac{1}{3}\right), \mathbf{u}_1 = \left(\frac{2}{3}, \dots, \frac{2}{3}\right)$;
Set $x_1{}^p = |b^p{}_1 - a^p{}_1| l_1{}^p + a^p$ , $y_1{}^p = |b^p{}_1 - a^p{}_1| u_1{}^p + a^p, p = 1, \dots, n$;
Set $\delta_1 = \max\{|b^p{}_1 - a^p{}_1|\}, p = 1, \dots, n$;
Set $m = 2, f_{\min} = \min\{f(\mathbf{x}_1), f(\mathbf{y}_1)\}, x_{\min} = \arg \min_{x \in \{\mathbf{x}_1, \mathbf{y}_1\}} f(\mathbf{x})$ ;
while $pe > \epsilon_{\text{tol}}$ and $m < N_{f_{\max}}$ do
    See Eq (10) for $pe$ definition;
    Identify the index set $\mathbb{P}_k \subseteq \mathbb{I}_k$ of Pareto optimal hyperrectangles;
                                #see (Definitions 2.2 & 2.3) for Pareto
                                     Optimal hyperrectangle definition
    Set $\mathbb{I}_k = \mathbb{I}_k \backslash \{\mathbb{P}_k\}$ ;
    #Take a new sample for each hyperrectangles in $\mathbb{P}_k$;
    for $i \in \mathbb{P}_k$ do
        Select coordinate index which is the branching variable;
            $\text{cb} = \min \left\{ \arg \max_{p=1,\dots,n} (d^p{}_i = |b^p{}_i - a^p{}_i|) \right\}$
        Set $\mathbf{l}_{m+2} = \mathbf{l}_{m+1} = \mathbf{l}_i$ dan $\mathbf{u}_{m+2} = \mathbf{u}_{m+1} = \mathbf{u}_i$ ;      #copy old sample points
        if $l_i{}^p < u_i{}^p$ then                                       #take new sample points.
            $l_{m+2}^{\text{cb}} = l_i^{\text{cb}} + \frac{d_i^{\text{cb}}}{2}$;
            $u_{m+1}^{\text{cb}} = u_i^{\text{cb}} - \frac{d_i^{\text{cb}}}{2}$;
            Determine the points of $\mathbf{x}_{m+2}, \mathbf{y}_{m+1}$ and then evaluate $f(\mathbf{x}_{m+2})$ and $f(\mathbf{y}_{m+1})$;
        else
            $l_{m+1}^{\text{cb}} = l_i^{\text{cb}} - \frac{d_i^{\text{cb}}}{2}$;
            $u_{m+2}^{\text{cb}} = u_i^{\text{cb}} + \frac{d_i^{\text{cb}}}{2}$;
            Determine the points of $\mathbf{x}_{m+1}, \mathbf{y}_{m+2}$ and then evaluate $f(\mathbf{x}_{m+1})$ and $f(\mathbf{y}_{m+2})$;
        end
        Set $f_{\min_{m+1}} = \min\{f(\mathbf{x}_{m+1}), f(\mathbf{y}_{m+1})\}, f_{\min_{m+2}} = \min\{f(\mathbf{x}_{m+2}), f(\mathbf{y}_{m+2})\}$
        Set $f_{\min} = \min\{f_{\min}, f_{\min_{m+1}}, f_{\min_{m+2}}\}, x_{\min} = \arg \min_{x \in \{x_{\min}, x_{\min_{m+1}}, x_{\min_{m+2}}\}} f(\mathbf{x})$;
        Bisect hyperrectangles $\bar{D}_i$ into two hyperrectangles, $\bar{D}_{m+1}$ dan $\bar{D}_{m+2}$ ;
        Update $pe, \delta_{m+1}, \delta_{m+2}$ ;
    end
    Increase $k = k + 1$ and set $\mathbb{I}_k = \mathbb{I}_k \cup \{m + 1, m + 2\}$;
end
```

namely the hyperrectangle that satisfies:

$$\min\left\{f(\mathbf{l}_{j_k^{\max}}), f(\mathbf{u}_{i_k^{\max}})\right\} < \min\left\{f(\mathbf{l}_{i_k^{\max}}), f(\mathbf{u}_{i_k^{\max}})\right\} \quad (5)$$

where $i_k^{\max}, j_k^{\max} \in \mathbb{I}_k^{\max}$.

■

Furthermore, a convergence theorem is provided to guarantee that the PLOBi algorithm will not terminate before reaching its stopping criteria.

Theorem 3.2: Let D be a feasible region of objective function f. Then, for any point $x \in D$ and any $\epsilon > 0$, there exists an iteration number $k(\epsilon) \geq 1$ and a hyperrectangle index $i_{k(\epsilon)} \in \mathbb{I}_k(\epsilon)$ such that

$$\max\left\{\left\|\boldsymbol{l}_{i_{k(\epsilon)}} - \boldsymbol{x}\right\|, \left\|\boldsymbol{u}_{i_{k(\epsilon)}} - \boldsymbol{x}\right\|\right\} \leq \epsilon. \quad (6)$$

Proof: Let $\mathbb{I}_k^{\max}$ represent the set of the hyperrectangle indices with the largest size, $\delta_{\max}^k$, at the iteration k. Based on Lemma 3.1, there exist a hyperrectangle with index $i_k^{\max} \in \mathbb{I}_k^{\max}$ selected for further search, where

$$\delta_{\max}^k = \max\left\{\delta_i^k; i \in \mathbb{I}_k\right\} = \max\left\{\delta_i^k; i \in \mathbb{I}_k^{\max}\right\}. \quad (7)$$

Sequentially, each hyperrectangles $i_k^{\max} \in \mathbb{I}_k^{\max}$ undergoes bisection partitioning, resulting in half the volume of the previous hyperrectangles. After a finite number of iterations, $\mathbb{I}_k^{\max}$ becomes an empty set. This process is applied iteratively to the new group containing the largest hyperrectangles. Consequently, as the number of partitions increases, the size of the largest hyperrectangles decreases. In other words, for any $\epsilon > 0$, there exists a finite number of iterations $k(\epsilon) \geq 1$ such that after $k(\epsilon)$ iterations of the PLOBi algorithm, the largest hyperrectangles will have a size smaller than ϵ, i.e.,

$$\delta_{\max}^{k(\epsilon)} = \max_{p=1,\dots,n}\left|b_{i_{k(\epsilon)}^{\max}}^p - a_{i_{k(\epsilon)}^{\max}}^p\right| \leq \epsilon. \quad (8)$$

Table I
KEY CHARACTERISTICS OF HEDAR TEST FUNCTION

Problem no.	Problem name	Dimension n	Feasible region $D = [\mathbf{a},\mathbf{b}]$	No. of local optima	Optimum value, 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	$[-100, 110]^2$	Unimodal	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	$[-10, 10]^n$	Multimodal	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	Hartmann	3	$[0, 1]^3$	4	-3.86278
18	Hartmann	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	$[-10, 15]^n$	Unimodal	0.0
24	Michalewics	2	$[0, \pi]^2$	2!	-1.80130
25	Michalewics	5	$[0, \pi]^5$	5!	-4.68765
26	Michalewics	10	$[0, \pi]^5$	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	$[-5, 10]^n$	Multimodal	0.0
37,38,39	Schwefel	2,5,10	$[-500, 500]^n$	Multimodal	0.0
40	Shekel, $m = 5$	4	$[0, 4]^4$	5	-10.15320
41	Shekel, $m = 7$	4	$[0, 4]^4$	7	-10.40294
42	Shekel, $m = 10$	4	$[0, 4]^4$	10	-10.53641
43	Shubert	2	$[-10, 10]^2$	760	-186.73091
44,45,46	Sphere*	2,5,10	$[-5.12, 6.12]^n$	Unimodal	0.0
47,48,49	Sum squares*	2,5,10	$[-10, 15]^n$	Unimodal	0.0
50	Trid	6	$[-36, 36]^6$	Multimodal	-36.0
51	Trid	10	$[-100, 100]^{10}$	Multimodal	-210.0
52,53,54	Zakharov*	2,5,10	$[-5, 11]^n$	Multimodal	0.0

From Eq. (8), for any $x \in D, x$ is located in a hyperrectangle $i_{k(\epsilon)} \in \mathbb{I}_{k(\epsilon)}$, and its size does not exceed ϵ, i.e.,

$$\delta_i^{k(\epsilon)} \leq \delta_{\max}^{k(\epsilon)} \tag{9}$$

The two points $\mathbf{l}_{i_{k(\epsilon)}}$ and $\mathbf{u}_{i_{k(\epsilon)}}$ are also in the hyperrectangle $i_{k(\epsilon)}$, so

$$\max\left\{\left\|\mathbf{l}_{i_{k(\epsilon)}} - \mathbf{x}\right\|, \left\|\mathbf{u}_{i_{k(\epsilon)}} - \mathbf{x}\right\|\right\} \leq \delta_i^{k(\epsilon)}. \tag{10}$$

Thus, from Eqs. (8)-(10), Eq. (6) is obtained.

■

Theorem 3.2 guarantees that the PLOBi algorithm converges to any point in the feasible region, implying "everywhere dense" convergence. As the sampled points taken by the PLOBi algorithm form a dense subset of the hypercube, the PLOBi algorithm will converge to the global optimal function value, $f(x^*)$ provided that the function f is continuous in the neighbourhood of the global minimum, x^*. The formal statement is summarized in the following Corollary 3.3.

Corollary 3.3: Suppose that f is continuous in the neighbourhood of the global minimum x^*. The PLOBi algorithm can sample a point $\tilde{x} \in D$, such that for any $\epsilon > 0$, the following inequality holds:

$$|f(\tilde{x}) - f(x^*)| < \epsilon. \tag{11}$$

The complete procedure of the PLOBi algorithm is given in Algorithm 1. The algorithm can be equipped with various stopping criteria, specified as initial input values: error tolerance (ϵ_{tol}), the maximum limit of the function evaluation, $(N_{f_{\max}})$, or the maximum number of iterations, $(t_{\max})$. Upon reaching the stopping criteria, the PLOBi algorithm returns the current best function value, $f_{\min}$, and solution point, $x_{\min}$ together with algorithmic performance measures: the number of function evaluations (N_f), and execution time (t).

IV. NUMERICAL RESULTS

In this section, the performance of the PLOBi algorithm is tested by numerical experiments conducted on the Hedar test set function, encompassing 54 problems with dimensions ranging from 2 to 10. The characteristics of the test problems used are detailed in Table I. Several problems required modifications to their feasible regions due to the global minimum point being the initial point in the initial sampling. Modified problems are denoted by the asterisk symbol (*).

In measuring the performance and efficiency of PLOBi algorithm, it is compared with other DIRECT-type algorithms. The PLOBi and BIRECT algorithms have similar sampling and partitioning procedures. Each hyperrectangle is partitioned into two equal sizes, and two sample points are taken on the main diagonal. The difference between the two lies in the technique for hyperrectangle selection and the size of the hyperrectangle used. The PLOBi algorithm uses the Pareto approach for optimal hyperrectangle selection and uses the infinity norm to measure the size of the hyperrectangle. By contrast, the BIRECT algorithm uses the same approach as

Table II
COMPARISON BETWEEN PLOBI, BIRECT-(NEW), AND DIRECT ALGORITHMS IN THE HEDAR TEST SET PROBLEMS WHEN $pe = 10^{-4}$

Problem	PLOBi		BIRECT-(New)		DIRECT	
No.	$f_{\min}$	N_f	$f_{\min}$	N_f	$f_{\min}$	N_f
1	2.54×10^{-5}	728	2.54×10^{-5}	**202**	7.53×10^{-5}	255
2	2.54×10^{-5}	3,058	2.54×10^{-5}	**1,256**	7.53×10^{-5}	8,845
3	2.54×10^{-5}	**25,632**	2.54×10^{-5}	45,128	7.53×10^{-5}	80,927
4	9.17×10^{-5}	**372**	9.17×10^{-5}	436	9.29×10^{-5}	655
5	4.01×10^{-5}	394	4.02×10^{-5}	468	3.09×10^{-5}	**327**
6	3.35×10^{-5}	390	3.35×10^{-5}	472	2.58×10^{-5}	**345**
7	3.66×10^{-5}	**402**	3.67×10^{-5}	474	8.21×10^{-5}	693
8	6.10×10^{-5}	**130**	6.10×10^{-5}	188	6.58×10^{-5}	295
9	0.39790	220	0.39790	242	0.39789	**195**
10	9.82×10^{-5}	**472**	9.82×10^{-5}	794	6.08×10^{-5}	6,585
11	9.07×10^{-5}	**318**	4.84×10^{-5}	722	6.25×10^{-5}	513
12	9.20×10^{-5}	5,788	7.15×10^{-5}	**4,060**	6.45×10^{-5}	19,661
13	0.03409	$> 100,000$	0.00024	$> 100,000$	0.00024	$> 100,000$
14	-0.99999	670	-0.99999	**558**	-0.99999	32,845
15	3.00025	202	3.00019	274	3.00009	**191**
16	7.76×10^{-7}	**2,362**	7.76×10^{-7}	4,982	4.84×10^{-6}	9,215
17	-3.86242	206	-3.86242	352	-3.86245	**199**
18	-3.32206	610	-3.32206	764	-3.32207	**571**
19	-1.03162	**146**	-1.03154	196	-1.03162	321
20	9.09×10^{-5}	130	9.09×10^{-5}	152	2.10×10^{-5}	**105**
21	1.83×10^{-5}	**484**	1.83×10^{-5}	968	3.65×10^{-5}	705
22	3.55×10^{-5}	**1,412**	3.55×10^{-5}	6,402	6.23×10^{-5}	5,589
23	2.71×10^{-5}	**68**	2.71×10^{-5}	90	3.81×10^{-5}	107
24	-1.80118	218	-1.80118	126	-1.80127	**69**
25	-4.64589	$> 100,000$	-4.68736	82,562	-4.68721	**13,537**
26	-8.1114	$> 100,000$	-7.32661	$> 100,000$	-7.87910	$> 100,000$
27	0.00729	$> 100,000$	0.00240	$> 100,000$	0.04325	$> 100,000$
28	8.34×10^{-5}	**738**	4.86×10^{-5}	2,114	9.02×10^{-5}	14,209
29	7.77×10^{-5}	**12,578**	9.87×10^{-5}	44,950	0.02142	$> 100,000$
30	9.00×10^{-5}	11,810	9.00×10^{-5}	**10,856**	0.00215	$> 100,000$
31	4.81×10^{-5}	242	4.81×10^{-5}	**180**	2.30×10^{-5}	987
32	1.18×10^{-5}	**1,126**	1.18×10^{-5}	1,162	4.97479	$> 100,000$
33	2.36×10^{-5}	**5,992**	2.36×10^{-5}	15,658	9.94967	$> 100,000$
34	9.65×10^{-5}	**168**	9.65×10^{-5}	180	2.30×10^{-5}	987
35	2.41×10^{-5}	**766**	2.41×10^{-5}	1,690	8.80×10^{-5}	20,025
36	5.42×10^{-5}	**1,938**	5.42×10^{-5}	9,100	8.29×10^{-5}	$> 100,000$
37	5.64×10^{-5}	300	3.09×10^{-5}	**236**	2.88×10^{-5}	255
38	6.41×10^{-5}	11,122	7.73×10^{-5}	**3,730**	7.21×10^{-5}	31,999
39	947.5068	$> 100,000$	0.08232	$> 100,000$	1187.6320	$> 100,000$
40	-10.15307	444	-10.15307	1,272	-10.15234	**155**
41	-10.40269	416	-10.40269	1,204	-10.40196	**145**
42	-10.53618	434	-10.53618	1,140	-10.53539	**145**
43	-186.72441	82,450	-186.72441	**1,780**	-186.72153	2,967
44	1.72×10^{-5}	**88**	1.15×10^{-5}	118	8.74×10^{-5}	209
45	2.87×10^{-5}	**350**	2.87×10^{-5}	602	9.39×10^{-5}	4,653
46	5.74×10^{-5}	**1,896**	5.74×10^{-5}	8,742	6.32×10^{-5}	99,123
47	7.94×10^{-6}	176	7.94×10^{-6}	226	3.52×10^{-5}	**107**
48	3.97×10^{-5}	**538**	3.97×10^{-5}	1,000	7.19×10^{-5}	833
49	9.11×10^{-6}	**2,100**	9.11×10^{-6}	5,538	7.76×10^{-5}	8,133
50	-49.99512	**920**	-49.99512	1,170	-49.99525	5,693
51	-204.67437	$> 100,000$	-209.98007	**32,170**	-209.98085	90,375
52	2.88×10^{-6}	**216**	2.88×10^{-6}	338	7.95×10^{-5}	237
53	1.01850	$> 100,000$	6.44×10^{-5}	**26,088**	9.71×10^{-5}	$> 100,000$
54	10.50758	$> 100,000$	9.43949	$> 100,000$	28.96394	$> 100,000$

the original DIRECT to select potentially optimal hyperrectangles (using a significant parameter). Both algorithms use the Euclidean norm to measure the hyperrectangle. The performance of the PLOBi algorithm is compared with those of the original DIRECT algorithm and the BIRECT-(New) algorithm in [26].

The numerical results of the algorithms are shown in Table II. The global optimization problem used is the set of Hedar test functions whose global minimum value (f^*) is known. Therefore, the stopping criteria for the algorithms are met when the point $\tilde{x}$ is obtained such that percent error (pe) where

$$pe = \begin{cases} \frac{f(\tilde{x})-f^*}{|f^*|}, & f^* \neq 0 \\ |f^*|, & f^* = 0 \end{cases} \tag{12}$$

is smaller than the tolerance value (ϵ_{tol}), or when the number of function evaluations exceeds the specified limit of 100,000. The best (smallest) number of function evaluations for each function is highlighted in bold font. If after reaching the specified function evaluation limit, the pe value has not met the tolerance value (ϵ_{tol}), the algorithm is considered unsuccessful in obtaining the global minimum point.

The numerical analysis for the Hedar test set in this section is grouped into three categories based on the number of problem dimensions: low-dimensional for $n = 2, 3,$

Table III
NUMBER OF FUNCTION EVALUTIONS USING THE PLOBI, THE BIRECT-(NEW) AND THE DIRECT ALGORITHMS

Problem	PLOBi			BIRECT-(New) [26]			DIRECT [2]		
no./pe	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}
1	728	1,502	2,282	**202**	**314**	**550**	225	443	655
2	3,058	5,540	8,456	**1,256**	**1,910**	**2,696**	8,845	11,289	14,619
3	**25,632**	**39,602**	**61,530**	45,128	70,090	92,328	80,927	$> 10^5$	$> 10^5$
4	**372**	984	1,508	436	**874**	**1,500**	655	1,143	1,823
5	394	568	782	468	690	994	**327**	**457**	**551**
6	390	446	782	472	532	988	**345**	**489**	**589**
7	**402**	**828**	**1,078**	474	1,202	1,524	693	1,073	1,645
8	**130**	**274**	**482**	188	410	744	295	511	917
9	220	488	$> 10^5$	242	7,742	$> 10^5$	**195**	**377**	**38,455**
10	**472**	**864**	**1,378**	794	1,282	1,968	6,585	18,261	24,485
11	**318**	**484**	**702**	722	898	1,334	481	597	1,143
12	5,788	6,652	7,744	**4,060**	**5,344**	**7,416**	18,237	19,407	23,605
13	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
14	670	670	**1,198**	**558**	**558**	$> 10^5$	32,859	59,347	$> 10^5$
15	202	372	**540**	274	2,332	$> 10^5$	**191**	**305**	10,437
16	**2,362**	**2,362**	**2,708**	5,106	5,106	5,464	9,215	9,341	9,341
17	206	**488**	$> 10^5$	352	26,940	$> 10^5$	**199**	4,165	**88,883**
18	610	**3,834**	**5,788**	764	$> 10^5$	$> 10^5$	**571**	$> 10^5$	$> 10^5$
19	**146**	**274**	**562**	196	$> 10^5$	$> 10^5$	293	997	54,487
20	130	212	396	152	266	532	**127**	**155**	**267**
21	**484**	**952**	**1,306**	968	1,900	2,416	705	1,021	1,921
22	**1,412**	**3,266**	**4,786**	6,402	14,274	18,676	5,589	10,431	18,475
23	**68**	**176**	**256**	90	238	334	107	209	391
24	218	316	338	126	458	458	**67**	**109**	**109**
25	$> 10^5$	$> 10^5$	$> 10^5$	82,562	**83,170**	$> 10^5$	**13,537**	$> 10^5$	$> 10^5$
26	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
27	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
28	**738**	**7,850**	**16,588**	2,114	9,470	20,052	13,675	67,515	$> 10^5$
29	**12,578**	$> 10^5$	$> 10^5$	44,950	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
30	11,810	18,260	28,860	**5,664**	**10,858**	**19,990**	$> 10^5$	$> 10^5$	$> 10^5$
31	242	378	648	**180**	**310**	**602**	987	1,181	1,565
32	**1,126**	**1,490**	**2,444**	1,162	1,588	2,850	$> 10^5$	$> 10^5$	$> 10^5$
33	**5,992**	**7,910**	**14,816**	15,658	18,820	30,220	$> 10^5$	$> 10^5$	$> 10^5$
34	**168**	322	496	180	**254**	**410**	1,621	1,913	3,005
35	**766**	**1,318**	**1,652**	1,690	2,786	3,318	19,693	24,681	35,575
36	**1,938**	**3,370**	**5,286**	9,100	14,268	20,534	$> 10^5$	$> 10^5$	$> 10^5$
37	300	$> 10^5$	$> 10^5$	**236**	$> 10^5$	$> 10^5$	255	**447**	**597**
38	11,122	$> 10^5$	$> 10^5$	**3,730**	$> 10^5$	$> 10^5$	27,543	**30,307**	**31,199**
39	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
40	444	838	$> 10^5$	1,272	6,482	$> 10^5$	**155**	**255**	$> 10^5$
41	416	$> 10^5$	$> 10^5$	1,204	**1,744**	$> 10^5$	**145**	4,875	$> 10^5$
42	434	**692**	**720**	1,140	1,616	$> 10^5$	**145**	4,939	$> 10^5$
43	82,450	82,644	83,014	**1,780**	**1,892**	$> 10^5$	2,967	3,867	**68,667**
44	**88**	**252**	**402**	118	374	594	209	417	633
45	**350**	**968**	**1,538**	602	1,924	3,202	4,653	10,583	20,123
46	**1,896**	**6,490**	**8,910**	8,742	26,996	35,756	99,123	$> 10^5$	$> 10^5$
47	176	252	452	226	332	624	**107**	**195**	**321**
48	**538**	**1,012**	**1,312**	1,000	1,894	2,522	833	1,489	2,463
49	**2,100**	**2,934**	**4,798**	5,538	7,172	12,076	7,795	14,691	22,651
50	**920**	**2,112**	**3,530**	1,170	$> 10^5$	$> 10^5$	4,897	$> 10^5$	$> 10^5$
51	$> 10^5$	$> 10^5$	$> 10^5$	**32,170**	**99,906**	$> 10^5$	90,375	$> 10^5$	$> 10^5$
52	**216**	350	**518**	338	546	802	237	**303**	653
53	$> 10^5$	$> 10^5$	$> 10^5$	**26,088**	**26,918**	27,876	$> 10^5$	$> 10^5$	$> 10^5$
54	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
Average results									
Overall	17,950	**24,282**	**31,133**	**15,149**	28,939	44,840	28,398	37,182	49,625
Unimodal	**638**	**1,437**	**2,104**	1,886	4,448	6,317	12,606	14,284	16,450
Multimodal	21,679	**31,060**	**39,148**	**17,802**	33,837	52,545	32,084	41,761	56,260
$n \leq 3$	3,943	8,463	17,355	**571**	10,967	31,194	2,290	**3,828**	**16,745**
$4 \leq n \leq 6$	17,847	**29,073**	**35,859**	**12,489**	29,415	47,070	27,409	47,086	65,972
$n > 6$	**45,963**	**55,298**	**58,344**	47,308	62,628	67,466	81,985	85,427	86,761
Failed	8	12	**15**	**5**	**11**	21	11	17	22

medium-dimensional for $n = 4, 5, 6$, and higher dimensional for $n > 6$. The 23 test problems are classified as low-dimensional. The PLOBi, BIRECT-(New) and DIRECT algorithms excel at 11, 6, and 6 test functions respectively. The PLOBi algorithm outperforms the BIRECT-(New) and DIRECT algorithms. This superiority is attributed to the smaller number of hyperrectangle groups formed based on Eq. (4) compared with using the Euclidean norm size, leading to a reduced number of function evaluations.

In the category of problems with medium-dimensional, there are 19 test problems, and the PLOBi, BIRECT-(New) and DIRECT algorithms excel at 8, 5, and 5 test functions respectively. However, the PLOBi and the DIRECT algorithms failed to reach the minimum value with a pre-determined pe in three test problems when the maximum number of

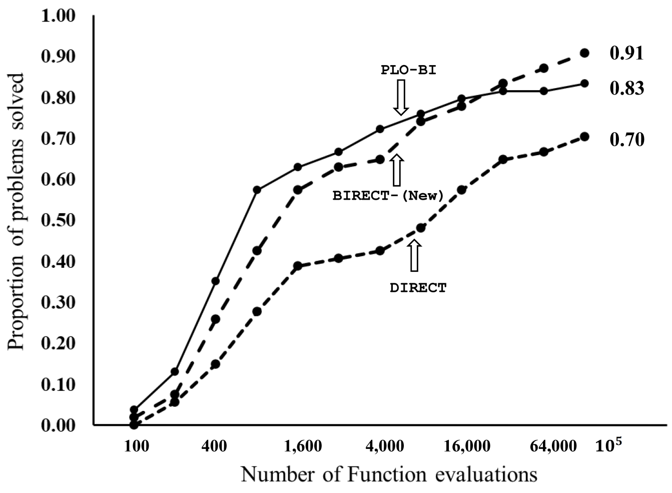

Figure 4. Operational characteristics of the PLOBi, BIRECT-(New), and DIRECT algorithms.

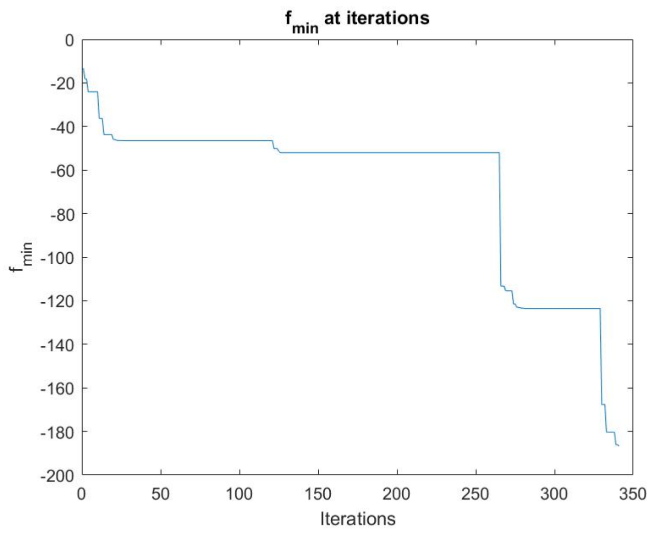

Figure 5. The historical curves of the minimum function values ($f_{\min}$) identified on the Shubert (2D) function.

function evaluations is limited to 100,000. The BIRECT-(New) algorithm failed to converge only on one test problem, the Perm function (No.27).

Moving on to the higher dimensional problems ($n > 6$) comprising 12 test problems, the DIRECT algorithm often fails to achieve convergence. It failed in seven test functions and only succeeded in five test functions. The BIRECT-(New) algorithm fails to converge on four test functions, and the PLOBi algorithm failed on five test functions. In the Powell function (No.28), only the BIRECT-(New) algorithm achieves convergence. These results indicate that for higher dimensional problems, when the PLOBi algorithm converges successfully, the resulting number of evaluations is 2 to 15 times better than those of the other two algorithms.

Fig. 4 illustrates the operational characteristics of the PLOBi, BIRECT-(New), and DIRECT algorithms. When the function evaluation is limited and very low ($N_{f_{\max}} < 1,600$), the proportion of Hedar test set problems solved by the PLOBi, BIRECT-(New), and DIRECT algorithms is $63\%, 57\%$, and 39%, respectively. As the budget for the function evaluation increases, specifically for ($N_{f_{\max}} < 100,000$), the number of problems solved by the BIRECT-(New) algorithm reaches 91%, followed by the PLOBi algorithm with 83%, and then the DIRECT algorithm with 70%.

One of the weaknesses of the original DIRECT method is its slow convergence with a high level of accuracy. We investigated the behavior of the PLOBi, the BIRECT-(New) [26] and the DIRECT [2] algorithms with three different pe values: $10^{-4}, 10^{-6}, 10^{-8}$, the results are shown in Table III.

Based on Table III, the overall average number of function evaluations of the PLOBi algorithm is best when the error rate (pe) is smaller, namely when $pe = 10^{-6}$ and $pe = 10^{-8}$. For low-dimensional problems ($n \leq 3$), the average performance of the PLOBi algorithm is not better than the other two algorithms. This is caused by one particular function whose function evaluation is excessive, namely the Shubert function. Based on Fig. 5, the horizontal curve shows no improvement in the $f_{\min}$ value after several iterations. Over-function evaluation by the PLOBi algorithm on the Shubert function occurs because the algorithm is stuck at a local minimum value. This can happen because the PLOBi algorithm does not use parameter such as the DIRECT and BIRECT-(New) algorithms to prevent searches that are too local. However, without using parameters it does not affect the performance of the PLOBi algorithm on other test functions.

For higher-dimensional problems ($n \geq 4$), the average performance of the PLOBi algorithm is 1.5 times to 1.8 times superior to that of the DIRECT algorithm for each pe value. In other words, to obtain solutions with high accuracy levels, the performance of the PLOBi algorithm is better than the DIRECT algorithm. Additionally, the failure rate at each pe value for the PLOBi algorithm is also lower than the DIRECT algorithm. For a sufficiently small pe level, specifically $pe = 10^{-8}$, the PLOBi algorithm fails in 15 test problems, compared to the failure of the BIRECT-(New) and the DIRECT algorithms, which fail in 21 and 22 test problems, respectively. In the case of unimodal functions, the performance of the PLOBi algorithm is also the best among the three pe values compared to the other two algorithms. The performance of the PLOBi algorithm in multimodal function cases is also relatively good, only lagging behind when $pe = 10^{-4}$ compared to the BIRECT-(New) algorithm.

From Table III, the performance of the PLOBi and BIRECT-(New) algorithms is mostly better than the DIRECT algorithm. Furthermore, these two algorithms are compared on several other test functions. Table IV presents the characteristics of several test functions found in [27], and the numerical experiment results of the PLOBi and BIRECT-(New) algorithms are presented in Table V. Based on Table V, the PLOBi algorithm outperforms the BIRECT-(New) algorithm in almost all test problems with three different variations of pe.

V. Conclusion

A novel DIRECT-type algorithm is introduced to solve box-constrained global optimization problems. The algorithm incorporates the Pareto approach, utilizing non-dominated hyperrectangles for further search. Each selected hyperrectangle undergoes bisection, dividing it into two equal sizes, and two points on the main diagonal are sampled. The size of

Table IV
KEY CHARACTERISTICS OF SOME TEST FUNCTIONS USED IN [27].

Problem name	Dimension n	Feasible region $D = [\mathbf{a},\mathbf{b}]$	Type	No. of local optima	Optimum value, f^*
Bukin	2	$[-15, 5] \times [10, 15]$	Convex	Multimodal	0.0
Drop-Wave*	2	$[-5.12, 6.12]^2$	Non-convex	Multimodal	-1.0
Langermann	2	$[0, 10]^2$	Non-convex	Multimodal	-4.1558
McCormick	2	$[-1.5, 4] \times [-3, 4]$	Convex	Multimodal	-1.91322
Rotated-Ellipsoid*	2,5,10	$[-65.536, 75.536]^n$	Convex	Unimodal	0.0
Styblinski Tang	2	$[-5, 5]^2$	Non-convex	Multimodal	$-39.1661n$
Sum Powers*	2,5,10	$[-0.55, 1.45]^n$	Convex	Unimodal	0.0

Table V
NUMBER OF FUNCTION EVALUTIONS USING THE PLOBI, AND THE BIRECT-(NEW) ALGORITHMS WITH DIFFERENCE pe.

Problem name /	PLOBi			BIRECT-(New) [26]		
pe	10^{-4}	10^{-6}	10^{-8}	10^{-4}	10^{-6}	10^{-8}
Bukin	$> 100,000$	$> 100,000$	$> 100,000$	$> 100,000$	$> 100,000$	$> 100,000$
Drop-Wave*	362	406	536	**190**	**210**	**444**
Langermann	**334**	**434**	**434**	686	798	798
McCormick	**104**	**158**	**158**	130	224	224
Rotated-Ellipsoid* $n = 2$	**172**	**290**	**638**	218	384	946
Rotated-Ellipsoid* $n = 5$	**854**	**854**	**1,812**	1,448	1,448	3,276
Rotated-Ellipsoid* $n = 10$	**2,020**	**3,378**	**4,218**	4,594	7,958	10,248
Styblinski Tang	**20**	**118**	**118**	26	124	124
Sum Power* $n = 2$	**32**	**100**	**228**	42	132	330
Sum Power* $n = 5$	**94**	**286**	**628**	42	132	330
Sum Power* $n = 10$	**154**	**578**	**4,984**	618	73,302	$> 100,000$

the hyperrectangle is measured using an infinity norm, facilitating the formation of fewer groups of hyperrectangles. The convergence of this algorithm is guaranteed. The combination of these procedures results in a new algorithm called the PLOBi algorithm. The performance of the PLOBi algorithm is tested using numerical experiments. The numerical results show that the PLOBi algorithm is not only effective but also highly compatible and competitive when compared with the other two DIRECT-type algorithms.

REFERENCES

[1] S. Alarie, C. Audet, A. E. Gheribi, M. Kokkolaras, and S. Le Digabel, “Two decades of blackbox optimization applications,” *EURO Journal on Computational Optimization*, vol. 9, p. 100011, 2021.

[2] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, “Lipschitzian optimization without the lipschitz constant,” *Journal of Optimization Theory and Applications*, vol. 79, pp. 157–181, 1993.

[3] T. A. El-Mihoub, A. A. Hopgood, L. Nolle, and A. Battersby, “Hybrid genetic algorithms: A review.” *Engineering Letters*, vol. 13, no. 2, pp. 124–137, 2006.

[4] Y. Ghanou and G. Bencheikh, “Architecture optimization and training for the multilayer perceptron using ant system,” *IAENG International Journal of Computer Science*, vol. 43, no. 1, pp. 20–26, 2016.

[5] R. Benedetti, M. M. Dickson, G. Espa, F. Pantalone, and F. Piersimoni, “A simulated annealing-based algorithm for selecting balanced samples,” *Computational Statistics*, vol. 37, no. 1, pp. 491–505, 2022.

[6] P. Hansen and B. Jaumard, *Lipschitz optimization*. Springer, 1995.

[7] S. Piyavskii, “An algorithm for finding the absolute extremum of a function,” *USSR Computational Mathematics and Mathematical Physics*, vol. 12, no. 4, pp. 57–67, 1972.

[8] B. O. Shubert, “A sequential method seeking the global maximum of a function,” *SIAM Journal on Numerical Analysis*, vol. 9, no. 3, pp. 379–388, 1972.

[9] R. Pandiya and E. Iryanti, “Stretching function technique based on the filled function algorithm for solving unconstrained global optimization problem: The univariate case.” *IAENG International Journal of Applied Mathematics*, vol. 53, no. 4, pp. 1461–1470, 2023.

[10] D. R. Jones and J. R. Martins, “The direct algorithm: 25 years later,” *Journal of Global Optimization*, vol. 79, no. 3, pp. 521–566, 2021.

[11] D. R. Jones, “Direct global optimization algorithm,” *Encyclopedia of Optimization*, pp. 431–440, 2001.

[12] L. M. Rios and N. V. Sahinidis, “Derivative-free optimization: a review of algorithms and comparison of software implementations,” *Journal of Global Optimization*, vol. 56, pp. 1247–1293, 2013.

[13] Q. Liu, J. Zeng, and G. Yang, “Mrdirect: a multilevel robust direct algorithm for global optimization problems,” *Journal of Global Optimization*, vol. 62, no. 2, pp. 205–227, 2015.

[14] Q. Liu, G. Yang, Z. Zhang, and J. Zeng, “Improving the convergence rate of the direct global optimization algorithm,” *Journal of Global Optimization*, vol. 67, pp. 851–872, 2017.

[15] L. Stripinis, R. Paulavičius, and J. Žilinskas, “Improved scheme for selection of potentially optimal hyper-rectangles in direct,” *Optimization Letters*, vol. 12, pp. 1699–1712, 2018.

[16] R. Paulavičius, Y. D. Sergeyev, D. E. Kvasov, and J. Žilinskas, “Globally-biased birect algorithm with local accelerators for expensive global optimization,” *Expert Systems with Applications*, vol. 144, p. 113052, 2020.

[17] Y. D. Sergeyev and D. E. Kvasov, “Global search based on efficient diagonal partitions and a set of lipschitz constants,” *SIAM Journal on Optimization*, vol. 16, no. 3, pp. 910–937, 2006.

[18] R. Paulavičius, L. Chiter, and J. Žilinskas, “Global optimization based on bisection of rectangles, function values at diagonals, and a set of lipschitz constants,” *Journal of Global Optimization*, vol. 71, pp. 5–20, 2018.

[19] J. M. Gablonsky, *Modifications of the DIRECT algorithm*. North Carolina State University, 2001.

[20] J. M. Gablonsky and C. T. Kelley, “A locally-biased form of the direct algorithm,” *Journal of Global Optimization*, vol. 21, pp. 27–37, 2001.

[21] L. Stripinis and R. Paulavičius, “Lipschitz-inspired halrect algorithm for derivative-free global optimization,” *Journal of Global Optimization*, pp. 1–31, 2023.

[22] J. Mockus, “On the pareto optimality in the context of lipschitzian optimization,” *Informatica*, vol. 22, no. 4, pp. 521–536, 2011.

[23] J. Mockus and R. Paulavičius, “On the reduced-set pareto-lipschitzian optimization,” *Computational Science and Techniques*, vol. 1, no. 2, pp. 184–192, 2013.

[24] J. Mockus, R. Paulavičius, D. Rusakevičius, D. Šešok, and J. Žilinskas, “Application of reduced-set pareto-lipschitzian optimization to truss optimization,” *Journal of Global Optimization*, vol. 67, pp. 425–450, 2017.

[25] A. Hedar, “Test functions for unconstrained global optimization,” http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm, 2005, accessed:23 July 2023.

[26] N.-E. Belkacem, L. Chiter, and M. Louaked, “A novel approach to enhance direct-type algorithms for hyper-rectangle identification,” *Mathematics*, vol. 12, no. 2, p. 283, 2024.

[27] S. Surjanovic and D. Bingham, “Virtual library of simulation experiments: Test functions and datasets,” https://www.sfu.ca/~ssurjano/index.html, 2013, accessed:22 April 2024.

Mira Mustika earned her Bachelor's degree from the Universitas Lampung, Lampung, Indonesia, in 2012. She obtained her Master's degree in Mathematics from Universitas Gadjah Mada, Yogyakarta, Indonesia, in 2016. In 2018, she joined the Mathematics Study Program, Faculty of Science at Institut Teknologi Sumatera, Lampung, Indonesia, as a lecturer. Currently, she is a doctoral student in the Mathematics Department at Universitas Gadjah Mada, with research interests in Optimization and Applied Mathematics.

Salmah is a Professor in the Department of Mathematics at Universitas Gadjah Mada, Yogyakarta, Indonesia. She earned both her undergraduate degree in 1989 and her doctoral studies in 2006 from the same department. In 1995, she received her Master's degree from the Mathematics Department at Institut Teknologi Bandung in Bandung, Indonesia. Her research areas include dynamic games, control theory, optimization, and their applications.

Indarsih completed her higher education from the undergraduate to the doctoral level at the Department of Mathematics, Universitas Gadjah Mada, Yogyakarta, Indonesia, in 1997, 2002, and 2015, respectively. She is an associate professor in the same department, with research interests in fuzzy programming and operations research.