An International Journal of Optimization and Control: Theories & Applications
ISSN: 2146-0957 eISSN: 2146-5703
Vol.16, No.6, pp.2249-2272 (2026)
https://doi.org/10.36922/IJOCTA026260135

RESEARCH ARTICLE

A two-stage pareto selection strategy with compromise programming for the DIRECT algorithm

Mira Mustika[1], Salmah Salmah[2*], and Indarsih Indarsih[2]

[1]Department of Mathematics, Institut Teknologi Sumatera, Lampung, Indonesia
[2]Department of Mathematics, Universitas Gadjah Mada, Yogyakarta, Indonesia
mira.mustika@ma.itera.ac.id, syalmah@yahoo.com, indarsih@ugm.ac.id

ARTICLE INFO

Article history:
Received: June 26, 2026
Revised: August 25, 2026
Accepted: September 1, 2026
Published Online: September 21, 2026

Keywords:
Global optimization
DIRECT-type algorithm
Two-stage selection
Parameterized pareto approach
Compromise programming

AMS Classification 2010:
90C26;65K10;49M37

ABSTRACT

A novel DIRECT-type algorithm employing a parameterized Pareto approach is proposed for box-constrained optimization problems. The method features a two-stage Pareto selection process. The first stage focuses on global exploration, using a parameter to identify the smallest significant hyperrectangle, which helps avoid prolonged entrapment in local minima. The second stage emphasizes local exploitation to improve convergence speed. Instead of selecting all non-dominated hyperrectangles during this stage, the algorithm reduces then to three groups: smallest, medium, and largest. A key innovation is the introduction of the medium hyperrectangle selection procedure in this stage based on the Compromise Programming solution approach. In addition, the algorithm incorporates diagonal sampling and bisection procedures for hyperrectangles, with hyperrectangle size measured using the infinity norm rather than the Euclidean norm. The performance of the proposed algorithm is evaluated through numerical experiments on benchmark test problems and ten GKLS test classes (1000 test functions in total). Numerical results show that the proposed algorithm is competitive with, and generally outperforms, several existing DIRECT-type algorithms. Moreover, the results obtained onthe ten GKLS test classes further support the effectiveness of the proposed algorithm across all considered problem dimensions.

CC BY

1. Introduction

Global optimization is a critical field within many rapidly advancing disciplines and continues to evolve significantly. Mathematical models of real-world systems often give rise to global optimization problems, which encompass many applications. These applications span diverse fields, including finance, statistics, allocation and location problems, operations research, network and transportation issues, structural optimization, engineering design, database management, nuclear and mechanical design, chemical engineering, and molecular biology.[1]

Among these applications, many involve complex scenarios in which the objective function is a black box, implying that the function lacks an explicit form and that its evaluation can be computationally expensive or time-consuming.[2] Due to these challenges, extensive global research efforts are ongoing, with numerous scholarly references dedicated to addressing these issues. The derivative-free approach is employed to address the black-box function issue.

There are several derivative-free approaches for solving global optimization problems, including stochastic, surrogate-assisted, and deterministic methods. Stochastic approaches, such as Genetic Algorithms,[3] Simulated Annealing,[4] and Particle Swarm Optimization,[5] have been widely investigated for solving various optimization problems.

*Corresponding Author

These methods employ stochastic search mechanisms to explore the search space, and their performance can vary depending on the characteristics of the optimization problem.[6] In general, stochastic approaches do not guarantee convergence to the global optimum for every run.[7]

Surrogate-assisted optimization has also received increasing attention for solving computationally expensive optimization problems.[8] In many practical engineering problems, evaluating the objective function can require substantial computational resources, while traditional optimization algorithms may require a large number of function evaluations to obtain an optimal solution. To address this issue, surrogate-assisted evolutionary algorithms construct surrogate models to approximate the objective function and use these models to guide the optimization process, thereby reducing the number of expensive function evaluations.[9] The effectiveness of these approaches depends on the construction of the surrogate model and on the strategy used to manage and update the model during the optimization process.

In contrast to stochastic and surrogate-assisted approaches, deterministic approaches offer different advantages and trade-offs. Under appropriate assumptions and search procedures, deterministic methods can provide theoretical guarantees of convergence to the global optimum. One popular deterministic approach for solving black-box function problems is the DIRECT method introduced by Jones *et al.*[10] The DIRECT (Dividing RECTangle) method combines space partitioning strategies with global selection criteria to efficiently explore and exploit the search space. In each iteration, multiple search points are calculated simultaneously, employing a range of weights between local and global searches. The method relies on simple parameter settings, avoiding the intricate tuning typically required to balance local and global search, thus producing a reliable and user-friendly algorithm.

The DIRECT method has been successfully applied across various fields, significantly improving decision-making, technology, and overall system functionality.[11] In financial optimization, the DIRECT algorithm has been applied to optimal portfolio selection,[12] which aids in maximizing returns while minimizing risk. The DIRECT algorithm has been used in transportation to optimize strategic parameters that minimize fuel consumption,[13,14] thereby contributing to the development of safer and more efficient transportation systems. In addition, the DIRECT method has proven effective in other real-world problems, such as healthcare,[15] geophysics,[16] weather routing for ships,[17] energy,[18] thermohydrodynamic lubrication,[19] image registration,[20] and others.[21–24]

Despite its successful applications, the DIRECT method has limitations. Several researchers[25–27] have identified key weaknesses in the method. One major issue is that in multimodal and high-dimensional problems, the DIRECT method often requires numerous function evaluations to explore local minima, which can delay finding the global minimum. Additionally, the method is relatively slow in producing high-accuracy solutions because it explores many medium-sized hyperrectangles simultaneously, whereas local refinement is required.

Several researchers have proposed modifying the original DIRECT algorithm to address these shortcomings. First, different sampling methods have been suggested. Instead of taking a single sample at the center of each hyperrectangle, some researchers propose sampling multiple points on the diagonals,[28] at the vertices,[29,30] or at other locations within each hyperrectangle.[31]

Another critical modification involves the procedure for selecting potentially optimal hyperrectangles. Some researchers enhance the original DIRECT algorithm by integrating it with a local optimizer,[25,27,32] which has proven highly effective. Other researchers[33–36] use a Pareto-based approach for selecting potentially optimal hyperrectangles, moving away from the lower-right convex hull approach in the DIRECT method.[10] The Pareto-based method tends to select more hyperrectangles than the lower-right convex-hull procedure, which is advantageous as it allows for more parallel exploration. In addition, the Pareto approach can reduce the need for parameter settings used in the lower-right convex-hull procedure, thereby simplifying the selection process.

In particular, according to,[37] among non-hybridized DIRECT-type algorithms, the best performance is often demonstrated by the DIRECT-GL[35] and 1-DTC-GL[38] algorithms, which use a two-step Pareto approach (global-local). The performance of these algorithms often even surpasses that of hybridized algorithms, especially for multimodal and non-convex problems.

However, according to,[39] although the DIRECT-GL algorithm addresses many of the shortcomings of the original DIRECT method, Pareto selection is not without limitations. In particular, the Pareto-based selection itself does not require the parameter used in the original DIRECT selection criterion,[10] but consequently lacks an explicit mechanism to control excessive local exploitation.

To address these issues, we propose a novel DIRECT-type algorithm that maintains the advantages of the Pareto selection strategy while overcoming its limitations. Our algorithm incorporates diagonal sampling and bisection procedures for hyperrectangles, with the hyperrectangle size measured by the infinity norm rather than the Euclidean norm. In addition, a two-stage Pareto selection strategy is performed at each iteration. The first stage focuses on global exploration and introduces a parameterized criterion only for the smallest Pareto hyperrectangle. This criterion is used to determine whether the improvement achieved by the smallest hyperrectangle is sufficiently significant to justify further subdivision. By restricting the use of the parameter to only one hyperrectangle, the additional computational cost remains low while reducing the risk of excessive subdivision around local minima and preserving the global exploration capability of the Pareto approach.

The second stage focuses on local refinement to improve convergence speed. Since this stage emphasizes local refinement rather than global exploration, selecting all non-dominated hyperrectangles may lead to unnecessary subdivisions and increased computational cost without improving the effectiveness of the local search. Therefore, instead of using the entire set of non-dominated hyperrectangles, we retain only three representative ones: the smallest, the medium, and the largest. The smallest and largest hyperrectangles preserve local exploitation and global exploration, respectively, while the medium hyperrectangle provides a compromise between these two search tendencies. The introduction of the medium hyperrectangle selection procedure constitutes a key innovation of the proposed algorithm. To identify this hyperrectangle, we employ the Compromise Programming approach,[40–42] which simultaneously considers the distance to the current best point and the hyperrectangle size to obtain a compromise solution.

This paper is organized as follows: Section 2 provides preliminary information about DIRECT-type algorithms. Subsection 2.3 discusses the Pareto-Lipschitzian approach in diagonal sampling and bisection partitioning. A detailed explanation of our algorithm (referred to as the PLOPBi-GL algorithm) is presented in Section 3. Numerical experiments using the Hedar test set are detailed in Section 4. Finally, conclusions drawn from this study are outlined in Section 6.

2. Preliminaries

In this paper, the global optimization problem being minimized is:

$$\min_{x_1,\ldots,x_n} f(x_1,\ldots,x_n), \tag{1}$$

$$\text{subject to : } a_p \le x_p \le b_p, p = 1,\ldots,n. \tag{2}$$

The function $f(\mathbf{x})$ may exhibit nonlinearity, nonconvexity, and nondifferentiability. The function f in **Equation (1)** can also be a black-box function. The constraints in **Equation (2)** correspond to a hyperrectangle-shaped search space, $H = [\mathbf{a},\mathbf{b}] \subseteq \mathbb{R}^n$.

Problems (1) and (2) can be addressed using DIRECT-type algorithms. The DIRECT-type algorithm involves three main procedures: sampling, partitioning, and selection. We will next review the most relevant DIRECT-type algorithms, starting with the original DIRECT algorithm, followed by the algorithm with diagonal sampling schemes (BIRECT), and finally, the algorithm utilizing the Pareto approach.

2.1. Original DIRECT algorithm

Without loss of generality, at the initialization stage, the search space $H = [\mathbf{a},\mathbf{b}] \subseteq \mathbb{R}^n$ is normalized to a unit hypercube, $\bar{H} = [\mathbf{0},\mathbf{1}] \subseteq \mathbb{R}^n$. Next, the initial sample point is selected at the center of the hypercube, $\mathbf{c} = \left(\frac{1}{2},\ldots,\frac{1}{2}\right)$, and the objective function is evaluated at this point. When the objective function is evaluated at the center of the hyperrectangle, $\mathbf{c} \in \bar{H}$, it corresponds to the center of the original domain $\mathbf{x} \in H$, i.e.,

$$f(\mathbf{c}) \equiv f(\mathbf{x}), \tag{3}$$

where

$$x_p = (b_p - a_p)c_p + a_p, p = 1,\ldots,n. \tag{4}$$

At this stage, trivially, the hypercube is selected as the potentially optimal hyperrectangle. Subsequently, selecting POHs in the BIRECT algorithm follows Definition 1 as outlined below.

Definition 1. *(Potentially optimal hyperrectangle) Suppose that in the most recent iteration, the search space $\bar{H} = [\boldsymbol{0},\boldsymbol{1}] \subseteq \mathbb{R}^n$ has been partitioned into hyperrectangles $\bar{H}^i = [\boldsymbol{a}^i,\boldsymbol{b}^i] \subseteq \bar{H}$ where $i \in \mathbb{I}_k$ represents the index of the hyperrectangles formed after k iterations. Let $\boldsymbol{c}^i$ represent the center point and let δ^i denote the size of the hyperrectangle $\bar{H}^i$. Let $\epsilon > 0$ and $f_{\min}$ denote the best currently known function value. A hyperrectangle $\bar{H}^j$ where $j \in \mathbb{I}_k$ is said to be potentially optimal if there exists some positive Lipschitz (rate-of-change) constant $\tilde{L} > 0$ such that*

$$f(\boldsymbol{c}^j) - \tilde{L}\delta^j \le f(\boldsymbol{c}^i) - \tilde{L}\delta^i, \forall i \in \mathbb{I}_k \tag{5}$$

$$f(\boldsymbol{c}^j) - \tilde{L}\delta^j \leq f_{\min} - \epsilon|f_{\min}|, \tag{6}$$

where the size of the hyperrectangle is denoted by

$$\delta^i = \frac{1}{2}\|\mathbf{b}^i - \mathbf{a}^i\|. \tag{7}$$

Suppose each hyperrectangle formed in an iteration is represented in a two-dimensional plane, where the horizontal axis denotes the size of the hyperrectangle and the vertical axis represents the function values at the sampled point. The selected hyperrectangle based on Definition 1 corresponds to the lower-right convex hull of the points. Subsequently, new sample points are taken from the selected hyperrectangle.

Let $\mathbf{c}^0$ denote the center point of a selected hyperrectangle, and let $Q = 1, 2, \ldots, n$ represent the set of coordinate indices or dimensions. The first step in the sampling and partitioning process is identifying the subset $\tilde{Q} \subset Q$ corresponding to the dimensions with the largest side length. Let d be one-third of this largest side length. In the second step, new sample points are taken at $\mathbf{c} \pm de_q$ for all $q \in \tilde{Q}$, where e_q is the unit vector in the q-th dimension. In **Figure 1a**, this means sampling points to the left, to the right, above, and below the center point, $\mathbf{c}^0$. These newly sampled points are depicted as red dots, with numbers above them that indicate the function's value.

Subsequently, the hyperrectangle containing $\mathbf{c}$ is partitioned into three sections along the dimensions in Q. Let w_q represent the best function value along the q-th dimension, defined as

$$w_q = \{f(\mathbf{c} + de_q), f(\mathbf{c} - de_q)\}. \tag{8}$$

In minimization cases, partitioning begins with the dimension having the smallest w_q value and proceeds to dimensions with higher w_q values. Based on **Figure 1**, the values $w_1 = \min\{60.65, 39.06\} = 39.06$, and $w_2 = \min\{60.75, 37.75\} = 37.75$ are obtained. Because $w_2 < w_1$, the first partition is performed along the coordinate $q = 2$ (vertical axis), and three new rectangles are obtained as in **Figure 1b**. Subsequently, the rectangle containing the center point is partitioned into three parts along the coordinate $q = 1$ (horizontal axis), as in **Figure 1c**.

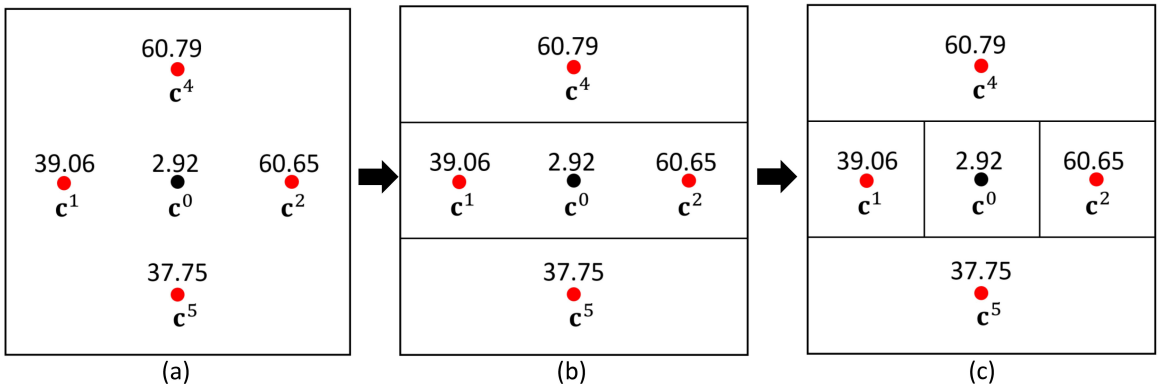


Figure 1. Sampling and partitioning rectangles using the DIRECT algorithm on the 2D Griewank function

2.2. Bisection rectangle (BIRECT) algorithm

Paulavičius *et al.*[28] proposed an innovative version of the DIRECT algorithm, referred to as the BIRECT algorithm, which employs a bisection strategy for rectangles instead of trisection and samples two points along the main diagonal of each rectangle. In the initialization step, two starting points located on the main diagonal are chosen after the initial search space is normalized to the unit hypercube.

Let these points be denoted by $\mathbf{l} = (l_1, \ldots, l_n) = \left(\frac{1}{3}, \ldots, \frac{1}{3}\right)$ and $\mathbf{u} = (u_1, \ldots, u_n) = \left(\frac{2}{3}, \ldots, \frac{2}{3}\right)$. At this stage, trivially, the hypercube is selected as the potentially optimal hyperrectangle. Subsequently, selecting POHs in the BIRECT algorithm resembles Definition 1. Since each hyperrectangle is represented by two sample points, $\forall i \in \mathbb{I}_k$ **Equations (5)** and **(6)** are replaced with:

$$\min\left\{f(\mathbf{l}^j), f(\mathbf{u}^j)\right\} - \tilde{L}\delta^j \leq \min\{f(\mathbf{l}^i), f(\mathbf{u}^i)\} - \tilde{L}\delta^i, \tag{9}$$

$$\min\left\{f(\mathbf{l}^j), f(\mathbf{u}^j)\right\} - \tilde{L}\delta^j \leq f_{\min} - \epsilon|f_{\min}|. \tag{10}$$

The size of the hyper rectangle is

$$\delta^i = \frac{2}{3}\|\mathbf{b}^i - \mathbf{a}^i\|. \tag{11}$$

Let $\bar{\mathbb{I}}_k$ denote the set of indices of the hyperrectangles formed at iteration k. Furthermore, let $\bar{\mathbb{I}}_k^P \subset \bar{\mathbb{I}}_k$ represent the set of indices of potentially optimal hyperrectangles. Bisection partitions are

applied to the hyperrectangles $\bar{D}^j, j \in \bar{\mathbb{I}}_k^P$, by dividing each along its longest side based on a single coordinate. If multiple coordinates share the longest side, the partition occurs along the coordinate with the smallest index. In other words, the hyperrectangle D^j, where the longest side measures δ_{br}^j, is bisected along the coordinate x_{br}, where

$$\text{br} = \min\left\{\arg\max_{p=1,\ldots,n}\left(d_p^j = |b_p^j - a_p^j|\right)\right\}. \quad (12)$$

Each recently partitioned hyperrectangle will contain one 'old' point from the previous iteration and one 'new' point. The two new points are:

$$\mathbf{l}^{i+2} = \left(l_1^i, \ldots, l_{\text{br}}^i \pm \frac{d_{\text{br}}^i}{2}, \ldots, l_n^i\right),$$

and

$$\mathbf{u}^{i+1} = \left(u_1^i, \ldots, u_{\text{br}}^i \pm \frac{d_{\text{br}}^i}{2}, \ldots, u_n^i\right).$$

Simultaneously, previous sampling points are reused, i.e., $\mathbf{l}^{i+1} = \mathbf{l}^i$ and $\mathbf{u}^{i+2} = \mathbf{u}^i$. The illustration of the sampling and bisection procedures in the BIRECT algorithm is presented in **Figure 2**.

The red points in **Figure 2** represent newly sampled points, whereas the black ones indicate where function values from earlier iterations are reused. The triangle symbols on each hyperrectangle indicate the best function values. The gray hyperrectangles correspond to the potentially optimal hyperrectangles selected at each iteration. Initially, the unit hypercube is trivially selected as the potentially optimal hyperrectangle (POH) (**Figure 2a**). Since both axes have the longest sides, partitioning is performed along the horizontal axis (coordinate $q = 1$). Furthermore, the POH in **Figure 2b** has its longest side along the vertical axis (coordinate $q = 2$). Therefore, partitioning is performed along the vertical axis, and new hyperrectangles are generated as shown in **Figure 2c**.

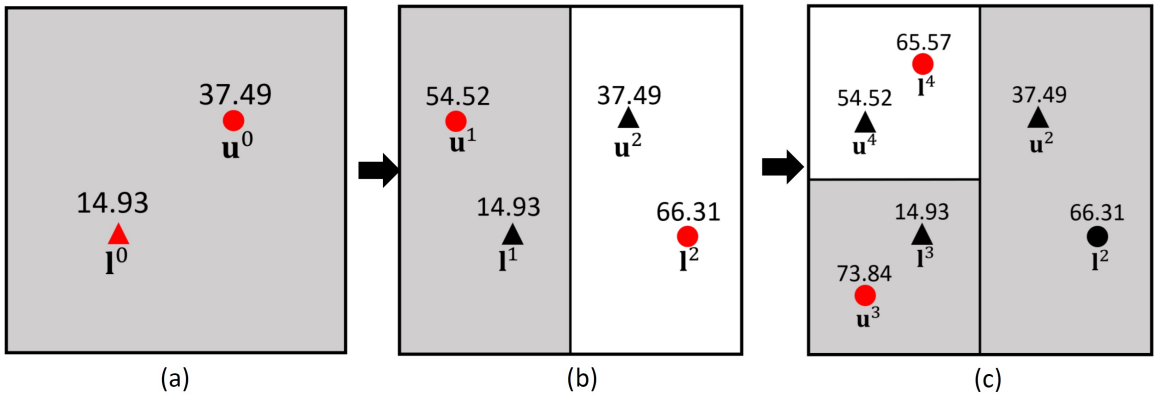


Figure 2. Sampling and partitioning rectangles using the BIRECT algorithm on the 2D Griewank function

2.3. Pareto-lipschitzian optimization in bisection partition and diagonal sampling (PLOBi algorithm)

Mockus *et al.*[40] introduced the concept of hyperrectangle dominance and Pareto optimal hyperrectangles, where each hyperrectangle is sampled at its center. Furthermore, Mustika *et al.*[36] defined hyperrectangle dominance in which each hyperrectangle is sampled at two points. In this study, the diagonal sampling procedure is employed, and the definition of hyperrectangle dominance from[36] is applied.

Definition 2. *Suppose that the search space $\bar{H} = [\mathbf{0}, \mathbf{1}] \subseteq \mathbb{R}^n$ has been partitioned into hyperrectangles $\bar{H}^i = [\boldsymbol{a}^i, \boldsymbol{b}^i] \subseteq \bar{H}$, where $i \in \mathbb{I}_k$ denotes the index of the hyperrectangles formed after k iterations. The points $\boldsymbol{l}^i$ and $\boldsymbol{u}^i$ represent two different sample points along the main diagonal of the hyperrectangle $\bar{H}^i$. Let F^i denote the best objective function value at the sample point for hyperrectangle $\bar{H}^i$, i.e., $F^i = \min\left\{f(\boldsymbol{l}^i), f(\boldsymbol{u}^i)\right\}$. Let δ^i denote the size of the hyperrectangle $\bar{H}^i$. Hyperrectangle $\bar{H}^j$ dominates hyperrectangle $\bar{H}^i, i, j \in \mathbb{I}_k$ (denoted $\bar{H}^j \prec \bar{H}^i$) if*

$$\delta^j \geq \delta^i \text{ and } F^j < F^i \quad (13)$$

or

$$\delta^j > \delta^i \text{ and } F^j \leq F^i \quad (14)$$

where

$$\delta^j = \max_{p=1,\ldots,n} d_p^j = \max_{p=1,\ldots,n} |\, b_p^j - a_p^j \,|. \quad (15)$$

Definition 3. *If there is no hyperrectangle $\bar{H}^j$ dominating $\bar{H}^i, i \in \mathbb{I}_k$, for all $j \in \mathbb{I}_k$, then the hyperrectangle $\bar{H}^i$ is said to be Pareto optimal.*

The size of hyperrectangle δ^j in **Equation (15)** is based on the infinity norm. This measure allows hyperrectangles that differ in characteristics to be classified into the same group. This results in fewer groups of hyperrectangles with identical sizes, even as the number of hyperrectangles within each group grows. In this study, the size of the hyperrectangle also corresponds to the infinity norm.

3. Two-stage parameterized pareto-lipschitzian optimization in diagonal sampling and bisection partition (PLOPBi-GL algorithm)

The concept of a non-dominated hyperrectangle, as given in Definitions 2 and 3, corresponds to the bi-objective optimization problem:

$$\min_{i\in\mathbb{I}_k} \left\{F^i, -\delta^i\right\}. \tag{16}$$

The first objective is to minimize the best objective function value at the sample points, and the second objective is to minimize the size of the hyperrectangle multiplied by -1 (which corresponds to maximizing the size of the hyperrectangle). The Pareto optimal hyperrectangles obtained by Definitions 2 and 3 will also correspond to the solutions of **Equation (16)**.

Unlike the potential optimality condition in the original DIRECT algorithm, these two criteria are treated separately rather than combined into a single scalar criterion. Thus, the objective function value and the hyperrectangle size are not directly combined in the selection process. Since the Pareto optimal hyperrectangles are determined from the dominance relation in Definition 2, we first examine whether this dominance relation is affected by a transformation of the objective function. The following proposition establishes this property.

Proposition 1. *Let $\mathbb{I}_k$ denote the set of indices of hyperrectangles formed after k iterations. Consider two hyperrectangles $\bar{H}^i, \bar{H}^j$, where $i, j \in \mathbb{I}_k$, and a transformed objective function*

$$\tilde{f}(\mathbf{x}) = af(\mathbf{x}) + b, \qquad a > 0. \tag{17}$$

Let $\tilde{F}^i$ and $\tilde{F}^j$ denote the corresponding best objective function values of $\bar{H}^i$ and $\bar{H}^j$, respectively. If $\bar{H}^j$ dominates $\bar{H}^i$ under f, then $\bar{H}^j$ also dominates $\bar{H}^i$ under $\tilde{f}$, and vice versa.

Proof. From the definition of F^i in Definition 2, the transformed best objective function value of hyperrectangle $\bar{H}^i$ is

$$\begin{aligned}\tilde{F}^i &= \min\left\{\tilde{f}(\mathbf{l}^i), \tilde{f}(\mathbf{u}^i)\right\}\\ &= \min\left\{af(\mathbf{l}^i)+b, af(\mathbf{u}^i)+b\right\}\\ &= a\min\left\{f(\mathbf{l}^i), f(\mathbf{u}^i)\right\}+b\\ &= aF^i + b.\end{aligned}$$

Similarly,

$$\tilde{F}^j = aF^j + b.$$

Since the transformation is applied only to the objective function, the size of each hyperrectangle remains unchanged, i.e.,

$$\tilde{\delta}^i = \delta^i, \qquad \tilde{\delta}^j = \delta^j. \tag{18}$$

Since $a > 0$, the ordering of the objective function values is preserved, such that

$$\begin{aligned}F^j < F^i &\iff aF^j + b < aF^i + b\\ F^j < F^i &\iff \tilde{F}^j < \tilde{F}^i,\end{aligned} \tag{19}$$

and similarly,

$$F^j \le F^i \iff \tilde{F}^j \le \tilde{F}^i. \tag{20}$$

From **Equations (18)**–**(20)**, both dominance conditions in Definition 2 are preserved under the transformation. □

The proposition shows that the dominance relation is invariant under a positive affine transformation of the objective function. We next consider the dominance relation among hyperrectangles of the same size. The following lemma shows that, within a group of hyperrectangles of equal size, a hyperrectangle with the smallest function value dominates the other hyperrectangles in the group.

Lemma 1. *Among a group of hyperrectangles of the same size u, the hyperrectangle with the smallest function value always dominates the others in the group.*

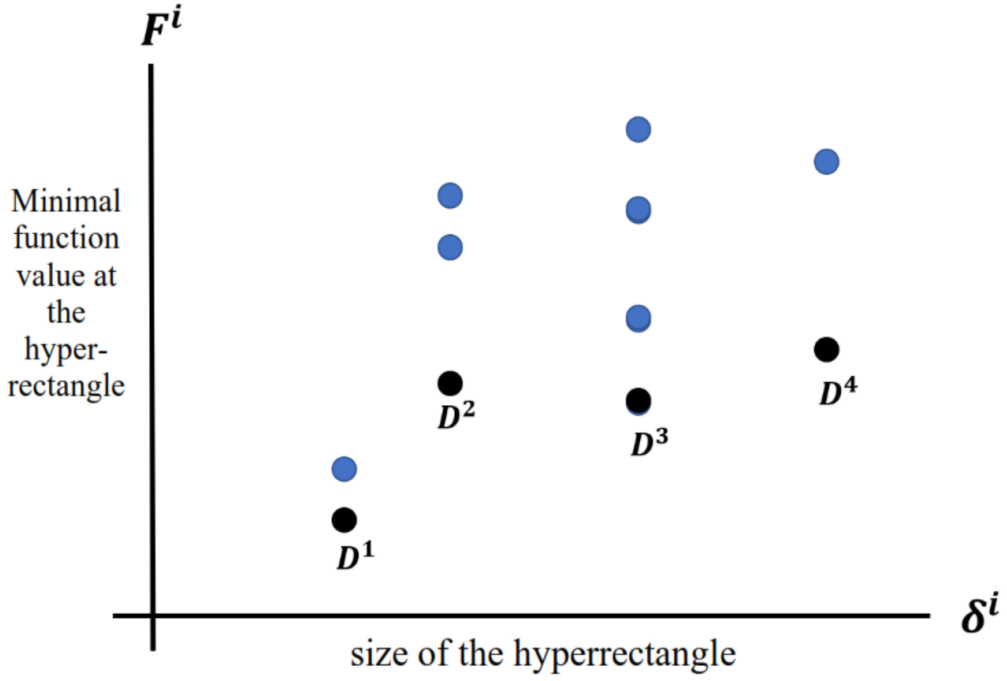
Proof. Let $\mathbb{I}_k$ denote the set of indices of hyperrectangles formed after k iterations. Let $A \subseteq \mathbb{I}_k$ denote a set of hyperrectangle indices with size u. Let the hyperrectangle H^a, where $a \in A$, be the one with the smallest function value among the hyperrectangles of size u. In other words,

$$F^a = \min_{i\in\mathbb{I}_k, \delta^i = u} F^i. \tag{21}$$

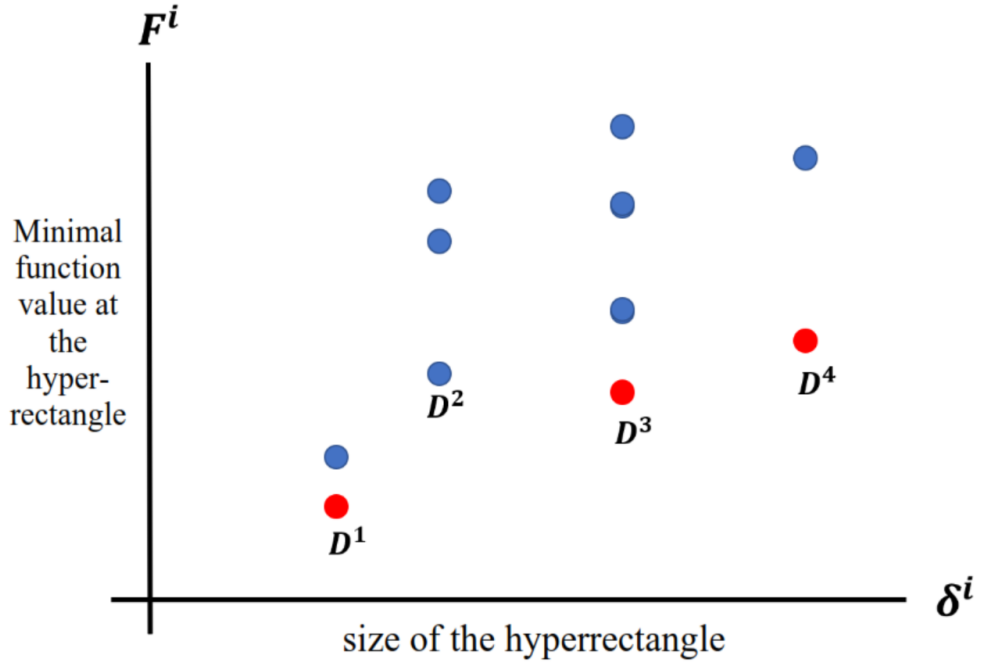
It will be shown that hyperrectangle H^a always dominates hyperrectangle H^i for all $i \in A$ with $F^i \neq F^a$. Let $i \in A$ be chosen arbitrarily, where $F^i \neq F^a$. From **Equation (13)**, we get $F^i > F^a$. Furthermore, from Definition 2, it can be said that hyperrectangle H^a dominates hyperrectangle H^i. □

Suppose the points in **Figure 3** represent hyperrectangles formed in an iteration. The black dots in **Figure 3a** correspond to the points outlined in Proposition 1. Not all of these points are Pareto optimal hyperrectangles. For instance, consider the point corresponding to hyperrectangle D^2, which is dominated by D^3. Therefore, hyperrectangle D^2 is not Pareto optimal. Only the points corresponding to hyperrectangles D^1, D^3, and D^4 are Pareto optimal, as indicated in **Figure 3b** (marked with red dots).

The following is an additional Lemma related to the Pareto optimal hyperrectangle.



(a) Identify the non-dominated hyperrectangle.



(b) Identify the Pareto optimal hyperrectangles.

Figure 3. Interpretation of the hyperrectangles formed in a two-dimensional plot

Lemma 2. *Let $\mathbb{I}_k$ denote the set of indices of hyperrectangles formed after k iterations. If the hyperrectangle H^i has the best function value and the largest size among the other hyperrectangles with the same function value, i.e.,*

$$\begin{aligned} F^i &= \min_{a \in \mathbb{I}_k} F^a = f^{\min}, \text{and} \\ \delta^i &= \max_{a \in \mathbb{I}_k,\, F^a = f^{\min}} \delta^a. \end{aligned} \tag{22}$$

then the hyperrectangle H^i is Pareto optimal.

Proof. Hyperrectangle H^i is said to be Pareto optimal if there is no hyperrectangle $H^j, j \in \mathbb{I}_k$, that dominates hyperrectangle H^i. The proof uses the contradiction. Suppose there is a hyperrectangle $H^j, j \in \mathbb{I}_k$, that dominates hyperrectangle H^i.

(1) Case $F^j = F^i, \forall j \in \mathbb{I}_k$. From **Equation (22)**, hyperrectangle H^i has the largest size. In other words, if $F^j = F^i$ for all $j \in \mathbb{I}_k$, then $\delta^i > \delta^j$. From **Equation (14)** in Definition 2, it follows that hyperrectangle H^i dominates hyperrectangle H^j, leading to a contradiction.

(2) Case $F^j \neq F^i, \forall j \in \mathbb{I}_k$. From **Equation (13)** in Definition 2, hyperrectangle H^j is said to dominate hyperrectangle H^i if $F^j < F^i$. Since hyperrectangle H^i has the best function value, and $F^j \neq F^i, \forall j \in \mathbb{I}_k$, it follows that $F^i < F^j$, leading to a contradiction.

□

3.1. The role of the parameter in the PLOBi algorithm

The PLOBi algorithm in[36] is parameter-free, which limits its ability to escape local minima effectively. For instance, in problems with numerous local minima, such as the Shubert-2D function, the PLOBi algorithm requires excessive evalutions, reaching 82,450 points for a tolerance level of 10^{-4}, which is 27 times that of the original DIRECT algorithm. For the 5-dimensional Michalewicz function, which has 5! or 120 local optima, the PLOBi algorithm fails to achieve convergence when the maximum number of evaluations is limited to 100,000. This may occur because the algorithm remains trapped at a local minimum. To address this critical limitation, the introduction of a parameter becomes essential, motivating the development of a new, more efficient algorithm. The role of this parameter is specified in Definition 4.

Definition 4. *Let $\mathbb{I}_k^P$ denote the set of indices of hyperrectangles satisfying Pareto optimality after k iterations. Let the hyperrectangle with index $z \in \mathbb{I}_k^P$ denote the smallest hyperrectangle with size δ^z. Let $f_k^{\min}$ denote the best function value obtained in iteration k, and let $\epsilon > 0$ be a positive constant. Let $\tilde{L} > 0$ denote the smallest slope of the line formed by (δ^z, F^z) and $(\delta^{p_k}, F^{p_k}), \forall p_k \in \mathbb{I}_k^P, \delta^{p_k} > \delta^z$, where $F^z = \min\{f(\boldsymbol{l}^z), f(\boldsymbol{u}^z)\}$. In other word,*

$$\tilde{L} = \min\left\{\frac{F^{p_k} - F^z}{*}\right\}, \forall p_k \in \mathbb{I}_k^P, \delta^{p_k} > \delta^z. \tag{23}$$

If $|\mathbb{I}_k^P| > 1$, hyperrectangle z is said to have a significant solution improvement if

$$F^z - \tilde{L}\delta^z \leq f_k^{\min} - \epsilon|f_k^{\min}|. \tag{24}$$

If $|\mathbb{I}_k^P| = 1$, hyperrectangle z is said to have a significant solution improvement.

The left-hand side of **Equation (24)** the lower bound of the function f on hyperrectangle z for some constant $\tilde{L} > 0$. The equation ensures that the lower bound of the smallest Pareto-optimal hyperrectangle must be significantly better than the current best solution, i.e., less than or equal to $f_k^{\min} - \epsilon|f_k^{\min}|$. This condition is necessary to prevent the algorithm from getting stuck in a local minimum when the improvement in the solution is insignificant.

From Definition 4, if hyperrectangle z experiences significant improvement, it will be included in the subsequent search. Conversely, the smallest Pareto optimal hyperrectangle z will be disregarded in the subsequent search if **Equation (24)** is not satisfied. In the PLOBi algorithm, all hyperrectangles identified as Pareto optimal are selected and subsequently partitioned for further searching. However, since the PLOBi algorithm does not utilize parameters, it may struggle to escape from local minima effectively. Therefore, integrating a parameter to filter the smallest rectangles is expected to help the algorithm avoid becoming trapped in local minima. Furthermore, the workflow of the PLOBi algorithm, when integrated with the parameter, is illustrated in the flow chart in **Figure 4**.

3.2. Two-stage procedure in the selection of pareto optimal hyperrectangles

The number of Pareto optimal hyper rectangles is greater than the number of potentially optimal hyperrectangles defined by.[10] On the one hand, this is advantageous because more hyperrectangles can be explored in parallel, allowing for more efficient search performance. On the other hand, a greater focus on global search can slow down the refinement of solutions to a high degree of accuracy, as more time is spent on medium-sized hyperrectangles rather than improving the local

minimum. To address this weakness,[34,40] reduced the set of Pareto optimal hyperrectangles by selecting only the largest and smallest hyperrectangles. The largest hyperrectangles are helpful for global search, while the smallest hyper rectangles are focused on local refinement. However, if the global minimum is located in a medium-sized rectangle (neither the smallest nor the largest), the reduced extent of global exploration may cause a delay in identifying the global minimum over many iterations.[39]

We employ a two-stage strategy to leverage the strengths of the Pareto approach while addressing its weaknesses. The first stage, called the global stage, focuses on global exploration by applying the procedure shown in **Figure 4**. In the second stage, called the local stage, the focus shifts to local refinement while still utilizing the Pareto approach.

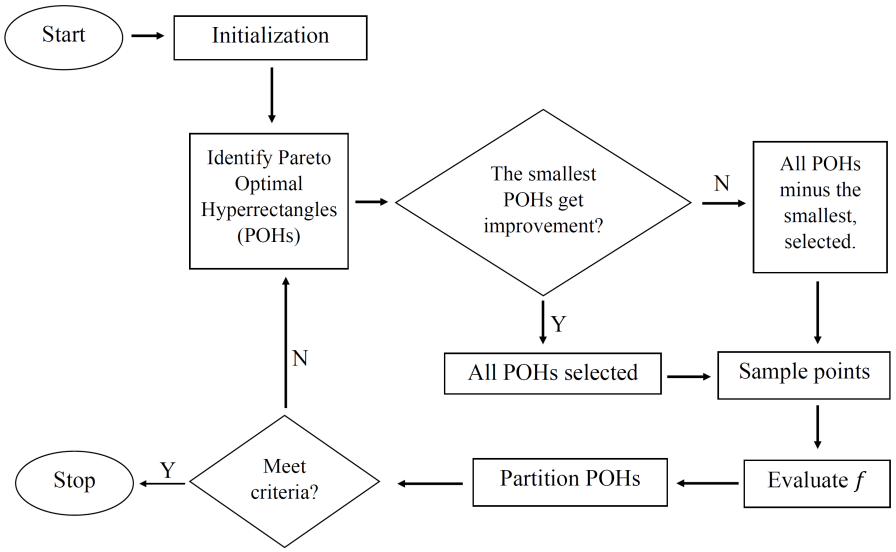


Figure 4. Flow chart of the PLOBi algorithm with parameter

The global stage in the PLOPBi-GL algorithm follows exactly the procedure of the PLOBi algorithm with the parameter, as shown in **Figure 4**. It begins by identifying the Pareto optimal hyperrectangles (POHs), according to Definitions 2 and 3. Next, the smallest POH is tested for its significance level based on Definition 4. If the smallest POH shows a significant improvement according to tolerance ϵ, then the hyperrectangle is selected. All hyperrectangles selected through this procedure are partitioned, and new sample points are taken before proceeding to the local stage.

We next examine the invariance of the global-stage selection under positive multiplicative scaling of the objective function.

Proposition 2. *Consider the positive multiplicative scaling of the objective function*

$$\tilde{f}(\mathbf{x}) = af(\mathbf{x}), \qquad a > 0. \tag{25}$$

If a hyperrectangle satisfies the significance condition in Definition 4 under f, then it also satisfies the same condition under $\tilde{f}$, and vice versa. Consequently, the hyperrectangles selected in the global stage remain unchanged under positive multiplicative scaling of the objective function.

Proof. At iteration k, let $\mathbb{I}_k^P$ denote the set of Pareto optimal hyperrectangles selected by the Pareto criteria. From Proposition 1, the Pareto-optimal set is unchanged under positive multiplicative scaling. Therefore, the smallest Pareto-optimal hyperrectangle index, z, and the set $\mathbb{I}_k^P$ remain unchanged. Since

$$\tilde{F}^i = \min\left\{\tilde{f}(\mathbf{l}^i), \tilde{f}(\mathbf{u}^i)\right\} = aF^i,$$

we have

$$\tilde{F}^z = aF^z.$$

Moreover, the size of each hyperrectangle is not affected by the transformation. Thus,

$$\tilde{\delta}^z = \delta^z.$$

Consequently, the slope defined in Definition 4 under the transformed objective function is

$$\begin{aligned}\tilde{L} &= \min_{\substack{p_k \in \mathbb{I}_k^P \\ \delta^{p_k} > \delta^z}} \left\{ \frac{\tilde{F}^{p_k} - \tilde{F}^z}{\tilde{\delta}^{p_k} - \tilde{\delta}^z} \right\} \\ &= \min_{\substack{p_k \in \mathbb{I}_k^P \\ \delta^{p_k} > \delta^z}} \left\{ \frac{aF^{p_k} - aF^z}{\delta^{p_k} - \delta^z} \right\} \\ &= a\tilde{L}.\end{aligned} \tag{26}$$

Furthermore, $f^k_{\min}$ is the best objective function value obtained in iteration k. Therefore, under the transformed objective function,

$$\tilde{f}^k_{\min} = a f^k_{\min}.$$

Therefore,

$$\tilde{F}^z - \tilde{L}\tilde{\delta}^z \leq \tilde{f}^k_{\min} - \epsilon \left|\tilde{f}^k_{\min}\right|$$
$$\iff aF^z - a\tilde{L}\delta^z \leq af^k_{\min} - \epsilon|af^k_{\min}|.$$

Because $a > 0$, we obtained:

$$\iff F^z - \tilde{L}\delta^z \leq f^k_{\min} - \epsilon|f^k_{\min}|. \quad (27)$$

Thus, the significance condition is satisfied under $\tilde{f}$ if and only if it is satisfied under f. Hence, the same hyperrectangles are selected in the global stage under both objective functions. □

3.3. Second phase: Local refinement

In the first stage, the algorithm focuses on global exploration to thoroughly explore the solution space. Therefore, the significance parameter ϵ is used to prevent the algorithm from getting stuck in local minima for too long. This is particularly important for problems with many local minima (i.e., multimodal problems).

In the second stage, the focus shifts to local refinement, while still employing the Pareto approach. Instead of using the bi-objective problem described in **Equation (16)**, this stage adopts a more locally focused approach, as described in,[35] for the PLOPBi-GL algorithm. This approach incorporates a distance function to the current minimum point, which aims to accelerate convergence toward better local solutions.

Let $\mathbf{s}_i$ denote the sample point of hyperrectangle i, and let $\mathbf{x}^{\min}$ be the current minimum point, i.e.,

$$\mathbf{x}^{\min} = \arg(f_{\min}) = \arg\min_{\mathbf{x}\in D}(f(\mathbf{x})). \quad (28)$$

The Euclidean distance between $\mathbf{s}_i$ and $\mathbf{x}_{\min}$ is expressed as:

$$d\left(\mathbf{x}^{\min}, \mathbf{s}_i\right) = \sqrt{\sum_{p=1}^{n}\left(x^{\min}_p - s^p_i\right)^2}. \quad (29)$$

In this section, the selected point $\mathbf{x}_{\min}$ point refers to the best point obtained at the end of the global phase. If, in an iteration, there is more than one $\mathbf{x}^{\min}$, the chosen $\mathbf{x}^{\min}$ is the one located in the hyperrectangle with the largest index, denoted by $\hat{\mathbf{x}}_{\min}$.

We present Lemma 3, which states that the minimum value of **Equation (29)** is attained at the point $\hat{\mathbf{x}}^{\min}$ itself.

Lemma 3. *Suppose $\hat{\mathbf{x}}_{\min}$ denotes the current minimum point. The minimum value of **Equation (29)** corresponds to the hyperrectangle that contains $\hat{\mathbf{x}}^{\min}$.*

Proof. Let the points $\mathbf{l}^i$ and $\mathbf{u}^i$ represent two sample points of hyperrectangle $\bar{H}^i$, $\forall i \in \mathbb{I}^g_k$, where $\mathbb{I}^g_k$ denotes the set of hyperrectangle indices formed at the end of the global stage (first stage). Since the point $\hat{\mathbf{x}}^{\min}$ is the best point obtained at the end of the global stage, there exists a hyperrectangle $\bar{H}^j$ with $j \in \mathbb{I}^g_k$ such that $\hat{\mathbf{x}}^{\min} = \mathbf{l}^j$ or $\hat{\mathbf{x}}^{\min} = \mathbf{u}^j$. Therefore, there exists a hyperrectangle $\bar{D}^j$ such that

$$s^j = \min\left\{d\left(\hat{\mathbf{x}}^{\min}, \mathbf{l}^j\right), d\left(\hat{\mathbf{x}}^{\min}, \mathbf{u}^j\right)\right\} = 0,$$

which is the hyperrectangle that contains $\hat{\mathbf{x}}^{\min}$. Since the distance function is non-negative,

$$s^j = 0 = \min_{i\in\mathbb{I}^g_k} s^i.$$

□

Let $\mathbb{I}^g_k$ be the set of hyperrectangle indices formed at the end of the global stage, and let $\hat{\mathbf{x}}_{\min}$ be the current minimum point obtained at that stage. Let $\mathbb{P}^l_k \subseteq \mathbb{I}^g_k$ represent the set of non-dominated hyperrectangle indices corresponding to the solution of a bi-objective function problem:

$$\min_{i\in\mathbb{I}_k}\left\{D^i, -\delta^i\right\}, \quad (30)$$

where

$$D^i = \min\left\{d\left(\hat{\mathbf{x}}^{\min}, \mathbf{l}^i\right), d\left(\hat{\mathbf{x}}^{\min}, \mathbf{u}^i\right)\right\}, \quad (31)$$

representing the shortest distance from the two sample points of hyperrectangle H^i to the current minimum point. To improve local solution refinement, not all hyper rectangles in $\tilde{\mathbb{P}}_k$ are selected, but rather reduced to three complementary categories: the smallest, largest, and medium-sized hyperrectangles. This reduction strategy is designed to maintain a balance between local exploitation and global exploration while retaining a compromise search direction. In particular, the smallest hyperrectangle represents an exploitation-oriented choice, whereas the largest hyperrectangle represents an exploration-oriented choice. The medium hyperrectangle, selected using the Compromise Programming (CP) approach, provides a compromise between these two search tendencies.

The smallest hyperrectangle corresponds to a solution of the single-objective problem $\min_{i\in\mathbb{I}^g_k}(D^i)$, which promotes local exploitation by favoring the hyperrectangle whose sample point is closest to the current minimum point. Furthermore, the largest hyperrectangle corresponds to

a solution of the single-objective problem $\min_{i\in\mathbb{I}_k^g}(-\delta^i)$, which preserves a global exploration tendency by favoring the largest hyperrectangle. The third choice is introduced to avoid relying exclusively on either of these two search tendencies. It is obtained through Compromise Programming and seeks a hyperrectangle that helps maintain a balance between local exploitation and global exploration.

In the PLOR algorithm introduced by,[40] the Pareto optimal hyperrectangle is reduced by selecting only the smallest and the largest ones. However, if the global optimum lies within a medium-sized hyperrectangle, this approach may slow down the solution search process. To address this issue, the present study introduces the concept of a medium hyperrectangle, inspired by the Compromise Programming (CP) solution.[41,42] Before defining a medium hyperrectangle, we first present the definition of the ideal and anti-ideal points, which are relevant in this context.

Definition 5. *The points $\boldsymbol{z}^* = (z_1, z_2)$ and $\boldsymbol{z}_* = (z_{1_*}, z_{2_*})$ are referred to as the ideal point and anti-ideal point, respectively, of the bi-objective problem (30), where*

$$z_1^* = \min_{i\in\mathbb{I}k}(D^i) \quad \textit{and} \quad z_2^* = \min_{i\in\mathbb{I}k}(-\delta^i), \tag{32}$$

and

$$z_{1_*} = \max_{i\in\mathbb{I}k}(D^i) \quad \textit{and} \quad z_{2_*} = \max_{i\in\mathbb{I}_k}(-\delta^i). \tag{33}$$

In the Compromise Programming method, solutions are selected based on their degree of proximity to the ideal point, either as feasible solutions or as Pareto optimal solutions. Furthermore, the definition of the degree of proximity relevant to this study is provided in Definition 6.

Definition 6. *Given the bi-objective problem (30), with the ideal and anti-ideal points as defined in Definition 5, the degree of closeness of the p-th objective function to its normalized ideal value (denoted as dk_p) is defined as follows:*

$$dk_1^i = \frac{|D^i - z_1^*|}{|z_{1_*} - z_1^*|} \textit{ and } dk_2^i = \frac{|-\delta^i - z_2^*|}{|z_{2_*} - z_2^*|}. \tag{34}$$

Next, we provide the definition of the medium hyperrectangle, which is related to the Compromise Programming solution, the solution that has the smallest degree of proximity (i.e., the closest distance) between the ideal point and the Pareto optimal solution in $\mathbb{P}_k^l$.

Definition 7. *Let $\mathbb{P}_k^l$ denote the set of Pareto-optimal hyperrectangle indices corresponding to the solution of Problem (30), with the ideal and anti-ideal points defined in Definition 5. Let the degree of closeness of each objective function to its ideal value be given in Definition 6. A hyperrectangle $j \in \mathbb{P}_k^l$ is said to be a medium hyperrectangle if it has the smallest degree of closeness, that is:*

$$dk^j_{\text{medium}} = \min_{i\in\mathbb{P}_k^l}(dk^i_{\text{ideal}}), \tag{35}$$

where

$$dk^i_{\text{ideal}} = \sqrt{(dk_1^i)^2 + (dk_2^i)^2}. \tag{36}$$

In the Pareto solution concept, when the optimized objective functions conflict with each other, the ideal solution is generally difficult to achieve. In the Compromise Programming (CP) method, the solution is selected based on its proximity to the ideal point, which represents the optimal values of all objective functions. Therefore, selecting the medium hyperrectangle using the Compromise Programming approach provides a compromise solution between the two objective functions considered simultaneously. In the proposed framework, the medium hyperrectangle selected by Compromise Programming is used together with the smallest and largest hyperrectangles. Thus, the three-category reduction strategy considers local exploitation, global exploration, and a compromise between the two to help maintain a balance between local and global search during the local refinement phase.

3.4. Convergence of the PLOPBi-GL algorithm

The main characteristic of the PLOPBi-GL algorithm is that a Pareto optimal hyperrectangle is identified twice in each iteration, namely during the global and local phases. The following Lemma guarantees the existence of at least one Pareto-optimal hyperrectangle, namely the one with the largest size.

Lemma 4. *Suppose that, in a given iteration, there are n_g groups of hyperrectangles formed. Among them, there always exists a hyperrectangle that is Pareto-optimal, specifically the one with the largest size.*

Proof. Let each group have sizes denoted by $u_1, u_2, \ldots, u_{n_g}$, where u_{n_g} is the largest size. Since each hyperrectangle is partitioned into two equal parts, we have

$$u_i = \frac{1}{2}u_{i+1}, \quad i = 1, \ldots, n_g - 1. \tag{37}$$

Let H^{u_i} denote a hyperrectangle that satisfies Lemma 1, for $i = 1, \ldots, n_g$. Since the hyperrectangle $H^{u_{n_g}}$ has the largest size, by Definition 2, there does not exist any hyperrectangle that dom-

inates $H^{u_{n_g}}$. In other words, the hyperrectangle $H^{u_{n_g}}$ is Pareto optimal. □

Each iteration of the PLOPBi-GL algorithm handles a group of hyperrectangles, where each group contains a finite number of hyperrectangles. Over successive iterations, the hyperrectangles within a group are partitioned into smaller groups. This procedure is repeated on the group containing the largest hyperrectangles, and continues until the size of the largest hyperrectangle falls below a predefined tolerance threshold. The PLOPBi-GL algorithm stops upon satisfaction of the stopping criterion. A formal statement is provided in the following theorem.

Theorem 1. *Let $H \subseteq \mathbb{R}^n$ be the feasible region for the objective function f. Suppose that H has been partitioned by bisection into several* hyperrectangles *H^i, $i \in \mathbb{I}_k$, where $\mathbb{I}_k$ denotes the index set of hyperrectangles formed at iteration k. For each point $\mathbf{x} \in H$ 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:*

$$\max\left\{\left\|\mathbf{l}^{i_{k(\epsilon)}} - \mathbf{x}\right\|, \left\|\mathbf{u}^{i_{k(\epsilon)}} - \mathbf{x}\right\|\right\} \leq \epsilon. \tag{38}$$

Proof. Based on Lemma 4, in both the global and local phases, there is always a Pareto optimal hyperrectangle selected, namely the one with the largest size, denoted by $\delta^{\max}$. One by one, the hyperrectangles are partitioned such that the volume of each resulting hyperrectangle is half the volume of its predecessor. Since each group of hyperrectangles contains a finite number of elements, after a finite number of iterations, the size of the largest hyperrectangle reduces to $\frac{1}{2}\delta^{\max}$. This process is then repeated on the new group containing the current largest hyperrectangle. As a consequence of this repeated partitioning, the size of the largest hyperrectangle continues to decrease. That is, for any $\epsilon > 0$, there exists a finite iteration number, $k(\epsilon) \geq 1$, such that after $k(\epsilon)$ iterations, the size of the largest hyperrectangle becomes smaller than ϵ, i.e.

$$\delta^{\max}_{k(\epsilon)} \leq \epsilon. \tag{39}$$

Therefore, any point $\mathbf{x} \in H$ lies within a hyperrectangle $H^{i_{k(\epsilon)}}$ such that

$$\delta^{i_{k(\epsilon)}} \leq \delta^{\max}_{k(\epsilon)}. \tag{40}$$

The hyperrectangle $H^{i_{k(\epsilon)}}$ also contains two diagonal sample points, namely $\mathbf{l}^{i_{k(\epsilon)}}$ and $\mathbf{u}^{i_{k(\epsilon)}}$. Therefore,

$$\max\left\{\left\|\mathbf{l}^{i_{k(\epsilon)}} - \mathbf{x}\right\|, \left\|\mathbf{u}^{i_{k(\epsilon)}} - \mathbf{x}\right\|\right\} \leq \delta^{i_{k(\epsilon)}}. \tag{41}$$

From **Equations (39)–(41)**, we obtain **Equation (38)**. □

Based on Theorem 1, for every point $\mathbf{x} \in H = [\mathbf{a}, \mathbf{b}] \subseteq \mathbb{R}^n$ and any tolerance $\epsilon > 0$, there exists a finite iteration at which a hyperrectangle is created whose diagonal sample points are within a distance of at most ϵ from $\mathbf{x}$. In particular, by choosing $\mathbf{x} = \mathbf{x}^*$, where $\mathbf{x}^*$ denotes a global minimizer of f, there exists a sample point $\mathbf{y}$ satisfying

$$\|\mathbf{x}^* - \mathbf{y}\| < \epsilon.$$

Therefore, if f is continuous in a neighborhood of the global minimizer $\mathbf{x}^*$, then the corresponding function value $f(\mathbf{y})$ can be made arbitrarily close to the global optimum $f(\mathbf{x}^*)$. The observation leads directly to the following corollary.

Corollary 1. *Suppose that the objective function f is continuous in a neighborhood of a global minimizer $\mathbf{x}^* \in H$. Then, for every $\xi > 0$, the PLOPBi-GL algorithm generates a sample point $\mathbf{y} \in H$ such that*

$$|f(\mathbf{y}) - f(\mathbf{x}^*)| < \xi.$$

The complete procedure of our algorithm is provided in **Algorithm 1**, which we refer to as the PLOPBi-GL algorithm. This algorithm represents a Pareto-Lipschitzian Optimization (PLO) approach, a bi-partition and bi-sampling scheme, and a global-local selection procedure. The algorithm can be configured with two primary input values: error tolerance (ϵ_{tol}) and maximum function evaluation limit ($N_{f_{\max}}$). Additionally, a maximum number of iterations ($t_{\max}$) can be included as input if needed. Upon meeting any of these termination criteria, the PLOPBi-GL algorithm outputs the current best function value ($f_{\min}$) and solution point ($\mathbf{x}^{\min}$), along with performance measures such as the number of function evaluations (N_f) and execution time (t).

4. Numerical experiments using hedar's test functions

This section presents the numerical evaluation of PLOPBi-GL using the Hedar test set. The experiments are conducted in two stages. First, a sensitivity analysis is performed to investigate the effect of the parameter ϵ and to determine an appropriate value for the subsequent experiments. Second, using the selected parameter value, the performance of PLOPBi-GL is compared with several existing DIRECT-type algorithms.

4.1. Hedar test set

In this section, we present a series of numerical experiments to evaluate the performance of the PLOPBi-GL algorithm. The numerical experiments were conducted on unconstrained global optimization problems as detailed in.[43] There are

54 global optimization problems with dimensions from 2 to 10, derived from 27 test functions, encompassing convex, non-convex, unimodal, and multimodal problems. **Table 1** provides a detailed description of the test function characteristics. An asterisk (*) symbol next to some test functions in the problem name column indicates modified problems. Modifications were made to the feasible region, as the initial sampling point was the global minimum.

Most problems in the Hedar test set exhibit multimodal characteristics. This characteristic makes them suitable for evaluating how effectively the proposed algorithm addresses one of the weaknesses of the DIRECT method: spending too many function evaluations on exploring suboptimal local minima, thus delaying the identification of the global minimum. Furthermore, the global minimum value for all test problems in the Hedar test set is known. Therefore, the algorithm stops when a point $\tilde{x}$ is found, such that the percent error (pe) is less than the error tolerance (ϵ_{tol}), where

$$pe = \begin{cases} \frac{f(\tilde{x})-f^*}{|f^*|}, & f^* \neq 0 \\ |f^*|, & f^* = 0 \end{cases} \tag{42}$$

or when the number of function evaluations reaches the specified maximum.

4.2. Sensitivity analysis and selection of ϵ

The parameter ϵ is introduced in the first stage of PLOPBi-GL to control the selection of the smallest Pareto-optimal hyperrectangles during the global exploration process. This mechanism is intended to prevent the search from becoming overly concentrated in a particular region and thereby reduce the risk of premature convergence to a local optimum. Therefore, the choice of ϵ may affect the balance between global exploration and computational effort.

To investigate the sensitivity of PLOPBi-GL to this parameter, six values of ϵ, namely $10^{-3}, 10^{-4}, 10^{-5}, 10^{-6}, 10^{-7}$, and 10^{-8}, were tested on the Hedar test set described in **Table 1**. The numerical experiments were conducted using two stopping criteria: a prescribed percent error $pe = 10^{-6}$, as defined in **Equation (42)**, and a maximum number of function evaluations of 25,000. The performance was evaluated based on the number of function evaluations required to satisfy the prescribed accuracy, subject to the maximum evaluation limit.

The results in **Table 2** show that $\epsilon = 10^{-3}$ achieves the lowest average number of function evaluations, with an average of 7765 evaluations. As ϵ decreases from 10^{-3} to 10^{-8}, the average number of function evaluations increases gradually, from 7765 to 7835. The small variation across the tested values indicates that the computational performance of PLOPBi-GL is relatively insensitive to the choice of ϵ within the tested range. This behavior is consistent with the role of ϵ as a selection parameter in the global exploration stage, which does not alter the objective function, sampling scheme, or partitioning rule. Based on these results, $\epsilon = 10^{-3}$ is selected for the subsequent numerical experiments.

Algorithm 1. The PLOPBi-GL Algorithm

Input: $\epsilon_{tol}, N_{f_{\max}}$;
Output: $f_{\min}, \mathbf{x}^{\min}$, performance measure;
The initial search space $H = [\mathbf{a},\mathbf{b}]$ is normalized to the unit hypercube $\bar{H} = [\mathbf{0},\mathbf{1}]$;
Initialization: $k = 1, \mathbb{I}_k = \{1\}, \mathbf{l}^1 = \left(\frac{1}{3}, \ldots, \frac{1}{3}\right), \mathbf{u}^1 = \left(\frac{2}{3}, \ldots, \frac{2}{3}\right)$;
Set $m = 2, f_{\min} = \min\left\{f(\mathbf{x}^1), f(\mathbf{y}^1)\right\}, \mathbf{x}^{\min} = \arg\min_{\mathbf{x}\in(\mathbf{x}^1,\mathbf{y}^1)} f(\mathbf{x})$;
while $pe > \epsilon_{tol}$ and $m < N_{f_{\max}}$ **do**
    See **Equation (??)** for the formula of pe;
    Identifying a set of Pareto optimal hyperrectangles (POHs) indices $\mathbb{P}_k \subseteq \mathbb{I}_k$ using Definitions 2 and 3;
    Identify the significance of the smallest hyperrectangle using Definition 4;
    Update $\mathbb{P}_k$ and set $\mathbb{I}_k = \mathbb{I}_k \setminus \{\mathbb{P}_k\}$;
    for all $i \in \mathbb{P}_k$ **do**
        Subdivide (bisect) all hyperrectangles in $\mathbb{P}_k$ and update $\mathbb{I}_k$;
        Evaluate f at diagonal sampling of the new hyperrectangles;
        Update $pe, f_{\min}, \mathbf{x}^{\min}$;
    end for
    for all $i \in \mathbb{I}_k$ **do**
        Calculate distance $d\left(\mathbf{x}^{\min}, \mathbf{l}^i\right)$ and $d\left(\mathbf{x}^{\min}, \mathbf{u}^i\right)$;
        Calculate $D_i = \min\left\{d\left(\mathbf{x}^{\min}, \mathbf{l}^i\right), d\left(\mathbf{x}^{\min}, \mathbf{u}^i\right)\right\}$;
    end for
    Identify a set of indices, $\tilde{\mathbb{P}}_k \subseteq \mathbb{I}_k$ representing POHs from Eq. (30);
    Determine $P_1 \subseteq \tilde{\mathbb{P}}_k$ where $P_1 = \arg\min_{i\in\tilde{\mathbb{P}}_k}(D^i)$;
    Determine $P_2 \subseteq \tilde{\mathbb{P}}_k$ where $P_2 = \arg\min_{i\in\tilde{\mathbb{P}}_k}(-\delta^i)$;
    Determine $P_3 \subseteq \tilde{\mathbb{P}}_k$ where P_3 satisfies **Equation (35)**;
    Set $\tilde{P}_k = P_1 \cup P_2 \cup P_3$;
    for all $i \in \tilde{P}_k$ **do**
        Subdivide (bisect) all hyperrectangles in $\tilde{P}_k$ and update $\mathbb{I}_k$;
        Evaluate f at diagonal sampling of the new hyperrectangles;
        Update $pe, f_{\min}, \mathbf{x}^{\min}$;
    end for
    Increase $k = k + 1$;
end while
Return: $f_{\min}, \mathbf{x}^{\min}, pe$ and other performance measures;

Table 1. Key characteristics of Hedar benchmark test functions

Problem no.	Problem name	Dimension n	Feasible region D = [a,b]	Number 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	$[-10, 10]^n$	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	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	$[-4, 5]^n$	Unimodal	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	$[-500, 500]^n$	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	$[-36, 36]^6$	Unimodal	-36.0
51	Trid	10	$[-100, 100]^{10}$	Unimodal	-210.0
52,53,54	Zakharov	2,5,10	$[-5, 11]^n$	Unimodal	0.0

Table 2. Sensitivity of PLOPBi-GL to ϵ

ϵ	Average number of function evaluations
10^{-3}	**7.765**
10^{-4}	7.770
10^{-5}	7.789
10^{-6}	7.802
10^{-7}	7.817
10^{-8}	7.835

4.3. Comparison with other DIRECT-type algorithms

In this study, the performance of the PLOPBi-GL algorithm was tested on four different ϵ_{tol} values: $10^{-4}, 10^{-6}, 10^{-8}$, and 10^{-10}. This approach allowed us to evaluate the effectiveness of the algorithms in overcoming the weakness of the original DIRECT method, which is slow in refining solutions with a high level of accuracy.

The performance of the PLOPBi-GL algorithm was compared with two other DIRECT-type algorithms: the DIRECT-GL algorithm and the original DIRECT algorithm. The selection procedures for promising hyperrectangles differ significantly between the PLOPBi-GL algorithm and the DIRECT-GL or original DIRECT algorithms. While PLOPBi-GL and DIRECT-GL utilize a two-step Pareto method, the PLOPBi-GL algorithm introduces a parameter. It reduces the number of promising hyperrectangles in the second step. Despite these differences, there is no notable variation in overall performance between

PLOPBi-GL and DIRECT-GL, nor in comparison with to the DIRECT algorithm. The time required to identify potentially optimal hyperrectangles is comparable across all three algorithms.

Table 3 summarizes the results of the numerical experiments comparing the PLOPBi-GL, DIRECT-GL, and DIRECT algorithms on the Hedar test set. The comparison is based on the number of function evaluations, with the best (i.e., smallest) value for each problem highlighted in bold. A star ($\star$) symbol indicates that all three algorithms failed to converge, i.e., failed to reach the prescribed error tolerance within the maximum allowed number of function evaluations, and that the corresponding algorithm achieved the best $f_{\min}$ among the three.

As shown in **Table 3**, the PLOPBi-GL algorithm requires a lower average number of function evaluations than the DIRECT-GL and the original DIRECT algorithms in terms of overall performance, unimodal problems, and both low-dimensional ($4 \leq n \leq 6$) and high-dimensional ($n > 6$) problems, across all four prescribed pe values ($10^{-4}, 10^{-6}, 10^{-8}, 10^{-10}$). In the multimodal category, PLOPBi-GL exhibits superior performance only for $pe = 10^{-4}$ and in the low-dimensional cases ($n = 4, 5, 6$). These results indicate that the PLOPBi-GL algorithm can mitigate the inefficiency of the original DIRECT method, particularly on problems where DIRECT requires a substantially larger number of function evaluations to achieve high-accuracy solutions.

The Rosenbrock function, also known as the banana function, is an unimodal problem where the global optimum lies in a long, narrow, parabolic-shaped flat valley.[44] In the Rosenbrock-10D problem (no. 36), the original DIRECT algorithm fails to achieve convergence, and the DIRECT-GL algorithm only succeeds when $pe = 10^{-4}$. In contrast, the PLOPBi-GL algorithm demonstrates notable performance on the Rosenbrock-10D problem. Not only does it achieve convergence for all four different pe values, but at $pe = 10^{-4}$, the total number of function evaluations required by the PLOPBi-GL algorithm is approximately 26 times lower than that required by the DIRECT-GL algorithm.

The PLOPBi-GL algorithm also demonstrates notable performance in the Rastrigin function problem. The Rastrigin function, which involves cosine modulation, causes the minima to be regularly distributed, making it highly multimodal. The DIRECT-GL and the original DIRECT algorithms fail to converge for the Rastrigin problem with dimensions $n = 5$ and $n = 10$. In contrast, the PLOPBi-GL algorithm successfully achieves convergence for all four different pe values, with a relatively low number of function evaluations, not exceeding 10,500.

The Friedman test was conducted to assess whether the three algorithms exhibit statistically significant differences in terms of their performance. For each test problem, the algorithms were ranked according to the number of function evaluations (NFE), with a smaller NFE receiving a better rank. When all three algorithms failed to converge within the prescribed maximum number of function evaluations, they were considered tied based on NFE. In this case, the ranking was determined based on the best objective value obtained, fmin, with a smaller value receiving a better rank. As shown in **Table 4**, PLOPBi-GL consistently achieves the lowest average rank across all error tolerances, with ranks ranging from 1.444 to 1.519. The Friedman test yields p-values below 0.05 for all error tolerances, indicating statistically significant differences among the three algorithms.

A total of 216 optimization problems were tested (54 Hedar test set problems, each with four different pe values). **Figure 5** presents the performance dominance of each algorithm. The PLOPBi-GL algorithm is the most dominant among the DIRECT-GL and original DIRECT algorithms, excelling in 59% (127/216) of the test problems. The second and third positions are held by the DIRECT-GL and the original DIRECT algorithms, with dominance rates of 17% (36/216) and 12% (26/216), respectively. In approximately 12% (27/216) of the problems, all three algorithms failed to achieve convergence.

5. Numerical experiments on GKLS benchmark problems

The Hedar test set used in the previous section provides an initial assessment of the PLOPBi-GL algorithm. To further evaluate its performance on more challenging global optimization problems, this section employs benchmark functions generated by the GKLS generator. Developed by Gaviano, Kvasov, Lera, and Sergeyev,[45] the GKLS generator provides three classes of test functions: non-differentiable, continuously differentiable, and twice continuously differentiable. Each class consists of 100 multiextremal multidimensional box-constrained optimization problems with known local and global optima, making them well suited for benchmarking global optimization algorithms.

Table 3. Number of function evalutions using the PLOPBi-GL, the DIRECT-GL and the DIRECT algorithms

Problem	**PLOPBi-GL**				**DIRECT-GL^{35}**				**DIRECT^{10}**			
	pe				**pe**				**pe**			
no.	**10^{-4}**	**10^{-6}**	**10^{-8}**	**10^{-10}**	**10^{-4}**	**10^{-6}**	**10^{-8}**	**10^{-10}**	**10^{-4}**	**10^{-6}**	**10^{-8}**	**10^{-10}**
1	472	792	1172	1644	1197	2123	3571	5415	**225**	**443**	**644**	**909**
2	**1854**	**3174**	**4866**	**6914**	19,403	35,175	55,843	84,979	8845	11,289	14,619	17,757
3	**16,944**	**28,670**	**38,562**	**50,498**	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	80,927	$> 10^5$	$> 10^5$	$> 10^5$
4	374	1050	1630	2284	**183**	**395**	**591**	**833**	655	1143	1823	2835
5	**284**	**392**	**516**	**758**	729	847	1115	1767	327	457	551	845
6	**286**	**318**	**518**	**762**	727	845	1113	1765	345	489	589	897
7	**314**	**702**	**864**	**1356**	685	1113	1665	2139	693	1073	1645	2099
8	**102**	**194**	**318**	**392**	345	509	831	1087	295	511	917	1295
9	210	430	$> 10^5$	$> 10^5$	333	579	**859**	**1239**	**195**	**377**	38.455	$> 10^5$
10	**392**	**648**	**964**	**1144**	1623	2809	3539	5371	6585	18,261	24,485	67,695
11	424	532	**648**	**1040**	**235**	**393**	823	1297	481	597	1143	1969
12	**9600**	**10,748**	**11,480**	**20,238**	13,109	16,501	22,951	31,213	18,237	19,407	23,065	32,229
13	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5\star$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
14	628	**628**	2648	2936	**495**	817	**1086**	**1679**	32,859	59,347	$> 10^5$	$> 10^5$
15	**158**	352	**538**	**750**	223	367	555	785	191	**305**	10,437	$> 10^5$
16	3062	3062	3272	3414	**2067**	**2375**	**2375**	**2799**	9215	9341	9341	9505
17	310	**616**	**688**	**688**	379	1049	1199	2431	**199**	4165	88,883	$> 10^5$
18	1362	**5668**	$> 10^5$	$> 10^5$	4793	8793	13.207	19.879	**571**	$> 10^5$	$> 10^5$	$> 10^5$
19	**206**	$> 10^5$	$> 10^5$	$> 10^5$	279	**485**	**657**	**1143**	293	997	54,487	$> 10^5$
20	**108**	176	292	**400**	189	263	459	581	127	**155**	**267**	401
21	**436**	**748**	**964**	**1452**	2349	4361	6329	10,149	705	1021	1921	2845
22	**2908**	**3836**	**4434**	**6028**	16,179	29,945	48,089	74.815	5589	10,431	18,475	28,461
23	**76**	**154**	**206**	**334**	101	211	357	557	107	209	391	935
24	182	302	328	328	129	235	235	235	**67**	**109**	**109**	**109**
25	35,692	36,120	36,948	36,986	**2445**	**4619**	**5575**	**5575**	14,077	$> 10^5$	$> 10^5$	$> 10^5$
26	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5\star$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
27	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5\star$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
28	**1002**	**11,848**	**20,218**	$> 10^5$	7045	24,591	85,235	$> 10^5$	13,675	67,515	$> 10^5$	$> 10^5$
29	**8500**	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
30	**27,788**	**33,972**	**47,376**	**58,486**	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
31	**182**	**268**	**434**	**534**	811	1109	1507	1803	987	1181	1565	1833
32	**828**	**1076**	**1720**	**2054**	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
33	**4426**	**6014**	**9750**	**12,218**	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
34	**166**	**256**	**380**	**446**	579	727	1143	1657	1621	1913	3005	4019
35	**730**	**1096**	**1300**	**1772**	25,395	38,633	72,735	86,043	19,693	24,681	35,575	41,687
36	**3624**	**4732**	**6088**	**6822**	95.405	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
37	**244**	$> 10^5$	$> 10^5$	$> 10^5$	659	971	1235	1709	255	**447**	**597**	**1195**
38	**5006**	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	27.543	**30.307**	**31.199**	**39.487**
39	$> 10^5\star$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
40	526	938	$> 10^5$	$> 10^5$	1227	2025	**3433**	**5209**	**155**	**255**	$> 10^5$	$> 10^5$
41	338	**642**	**1148**	**1148**	1141	2845	4741	6623	**145**	4875	$> 10^5$	$> 10^5$
42	338	**640**	$> 10^5$	$> 10^5$	1151	2871	**4749**	**7137**	**145**	4939	$> 10^5$	$> 10^5$
43	1600	1664	2082	$> 10^5$	**425**	**735**	**951**	**1341**	2967	3867	68.667	$> 10^5$
44	**82**	**212**	**310**	**310**	391	549	737	1103	209	417	633	1211
45	**334**	**826**	**1174**	**1796**	4357	8249	11,011	18,225	4653	10,583	20,123	44,099
46	**2664**	**4888**	**6818**	**10,202**	35,721	63,399	94,991	$> 10^5$	99,123	$> 10^5$	$> 10^5$	$> 10^5$
47	146	**194**	**316**	**468**	191	337	525	759	**107**	195	321	623
48	**510**	**844**	**1050**	**1494**	2919	4701	7523	11,031	833	1489	2463	3827
49	**2810**	**3284**	**4412**	**5716**	24,763	41,781	63,413	89,543	7795	14,691	22,651	34,735
50	**1224**	**2620**	**3338**	**3338**	7795	15,735	26,059	38,929	4897	$> 10^5$	$> 10^5$	$> 10^5$
51	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	**36.525**	$> 10^5\star$	$> 10^5$	$> 10^5$	90,375	$> 10^5$	$> 10^5$	$> 10^5$
52	**166**	**246**	**342**	**498**	345	413	889	1123	237	303	653	949
53	19,268	19,492	19,872	**20,228**	**6429**	**9967**	**17,665**	23,891	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$
54	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5\star$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$	$> 10^5$

(*Cont'd...*)

Table 3. (*Continued*)

Problem	PLOPBi-GL				DIRECT-GL[35]				DIRECT[10]			
	pe				pe				pe			
no.	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-4}	10^{-6}	10^{-8}	10^{-10}	10^{-4}	10^{-6}	10^{-8}	10^{-10}
Average results												
Overall	**14,053**	**22,131**	**30,370**	**34,590**	26,324	30,268	34,659	38,035	28,838	37,181	49,624	56,379
Unimodal	**16,735**	**22,008**	**22,775**	**27,385**	25,563	33,070	41,273	45,828	34,667	42,862	45,977	48,565
Multimodal	**12,347**	**22,209**	35,203	39,175	26,847	28,431	**30,111**	**32,676**	24,813	33,276	52,132	61,751
$n \leq 3$	**425**	9241	13,804	18,232	509	**759**	**1064**	**1533**	2290	3828	16.745	27.462
$4 \leq n \leq 6$	**10,907**	**17,426**	**34,338**	**39,845**	26,378	30,625	38,981	44,961	27,380	47,086	65,972	71,033
$n > 6$	**45,156**	**54,285**	**55,839**	**57,624**	75,761	86,260	92,,204	97,030	81,985	85,427	86,761	88,600
Failed	**6**	**10**	14	16	11	13	**13**	**15**	11	17	22	26

Table 4. Average Friedman ranks of the three algorithms across different error tolerances

Algorithm	10^{-4}	10^{-6}	10^{-8}	10^{-10}
PLOPBi-GL	**1.4444**	**1.4537**	**1.5185**	**1.4722**
DIRECT-GL	2.2963	2.1852	2.0370	2.0463
DIRECT	2.2592	2.3611	2.4444	2.4815
p-value	3.66×10^{-6}	3.71×10^{-6}	8.90×10^{-6}	9.78×10^{-7}

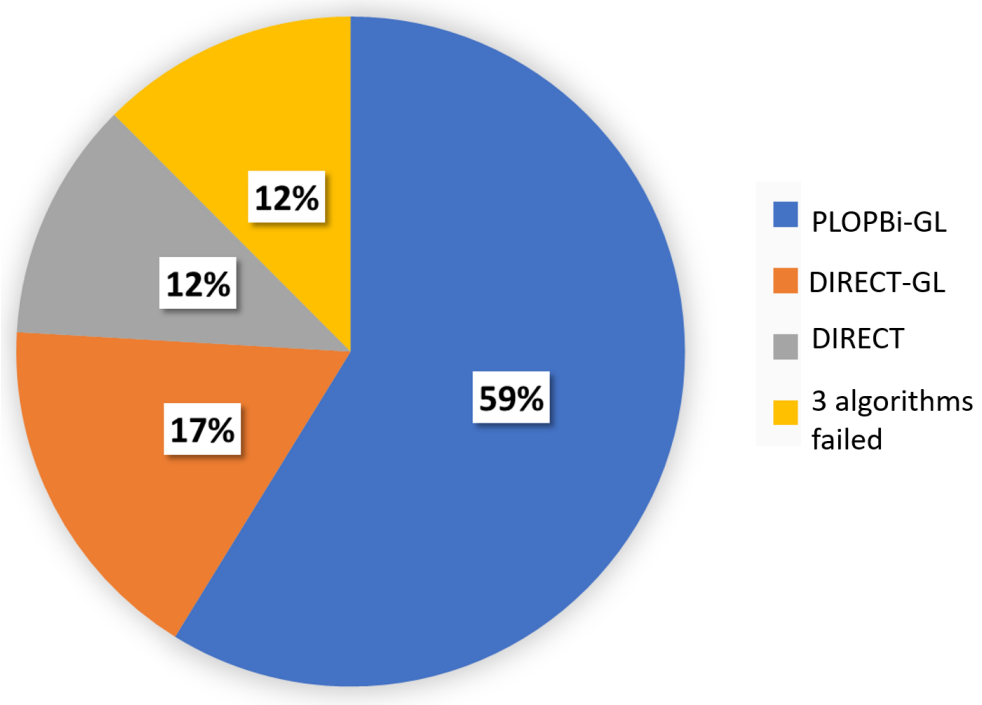


Figure 5. Performance of the PLOPBi-GL, DIRECT-GL, and DIRECT on the Hedar test functions with four *pe*

The test functions are generated by modifying a convex quadratic function in a systematic manner so that multiple local minima are introduced while preserving a known global optimum. Each problem class is characterized by several parameters, including the problem dimension (n), the number of local minima (m), the value of the global minimum (f^*), the radius of the attraction region associated with the global minimizer (r^*), and the distance between the global minimizer and the vertex of the underlying quadratic function (d). The remaining characteristics of the test functions, such as the locations and objective values of the minima and the corresponding attraction regions, are automatically determined

by the generator. Consequently, the GKLS generator provides a diverse collection of benchmark problems with controlled properties, enabling a systematic evaluation of global optimization algorithms. Furthermore, for a given set of parameter values, the generator always produces the same collection of 100 test functions, ensuring reproducibility of numerical experiments.

Table 5. GKLS benchmark classes used in the numerical experiments.

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

In the present study, the numerical experiments are conducted using the non-differentiable class of GKLS test functions. A total of 10 problem classes are considered, covering dimensions $n = 2$ to $n = 6$. For each dimension, two categories of problems, namely Simple and Hard, are generated, resulting in a total of 1000 test functions. The distinction between these categories is determined by the parameters r^* and d, which denote the radius of the attraction region of the global minimizer and the distance between the global minimizer and the vertex of the underlying quadratic function, respectively. In general, smaller values of r^* and larger values of d increase the difficulty of locating the global optimum. Consequently, the Hard classes are expected to be more challenging than the corresponding Simple classes.

The parameter settings used in this study are adopted from.[46] In particular, the number of local minima is fixed at $m = 10$, and the value of the global minimum is set to $f^* = -1$ for all problem classes. The parameter values defining the Simple and Hard classes are summarized in **Table 5**.

The performance of the proposed PLOPBi-GL algorithm is evaluated by comparing it with PLOBi,[36] DIRECT-GL,[35] and HALRECT[31] algorithms on the GKLS benchmark problems described above. HALRECT is included as a recent DIRECT-type algorithm that uses a hyperrectangle partitioning strategy based on halving, similar to PLOPBi-GL. Both algorithms therefore use bisection-based partitioning, while they differ in their sampling and selection strategies. A test problem is considered successfully solved when the percent error (pe) defined in **Equation (42)** satisfies $pe \leq 10^{-4}$. The maximum number of function evaluations is set to $n \times 10^4$, where n denotes the problem dimension. A problem is considered unsolved if the required accuracy is not achieved within the specified function-evaluation budget.

Figures 6 and **7** present the operational characteristics of PLOPBi-GL, PLOBi, DIRECT-GL, and HALRECT on the ten GKLS test classes. The figures show the cumulative number of solved problems as a function of the number of function evaluations. **Figure 6** presents the results for dimensions $n = 2$ to $n = 4$, while **Figure 7** presents the results for dimensions $n = 5$ and $n = 6$. A problem is counted as solved when the stopping criterion $pe \leq 10^{-4}$ is satisfied before reaching the maximum number of function evaluations. Thus, the operational characteristic curves provide a direct comparison of the efficiency of the four algorithms in solving the GKLS problems within the given function-evaluation budget. A curve that rises faster indicates that more problems are solved with fewer function evaluations.

The detailed numerical results are summarized in **Table 6**. The results show that PLOPBi-GL generally requires fewer function evaluations than PLOBi, DIRECT-GL, and HALRECT, especially for the first four classes. In terms of the average Friedman rank based on the number of function evaluations, PLOPBi-GL obtains the lowest rank in all ten GKLS classes. Its average rank over the ten classes is 2.074, compared with 2.830 for PLOBI, 2.524 for DIRECT-GL, and 2.572 for HALRECT. This indicates that PLOPBi-GL provides a more consistent performance in terms of the number of function evaluations.

The PLOPBi-GL algorithm successfully satisfies the stopping criterion for all problems in Classes 1–4, with no failed runs. Its advantage is also visible in these classes, where the average and median numbers of function evaluations are generally lower than those of the other algorithms. Overall, PLOPBi-GL successfully solves 86.3% of the 1000 test problems. This is slightly lower than DIRECT-GL (87.5%), but higher than HALRECT (84.8%) and PLOBI (60.6%). However, the number of failed runs increases for the more difficult classes, particularly Classes 6, 8, and 10. This shows that although PLOPBi-GL has a good average rank in terms of function evaluations, satisfying the stopping criterion becomes more challenging for the harder GKLS problems. This may be related to the higher difficulty of locating the global optimum with the required accuracy within the given evaluation budget.

Table 6. Numerical results on eight GKLS test classes described in **Table 5**

Class	PLOPBi-GL	PLOBi	DIRECT-GL	HALRECT
		Maximum number of function evaluations		
1	**532**	1226	521	701
2	**4148**	$> 20,000$ (11)	9531	7965
3	5754	$> 30,000$ (5)	**5227**	8289
4	29,360	$> 30,000$ (23)	29,247	**20,347**
5	$> 40,000$ **(4)**	$> 40,000$ (46)	$> 40,000$ (8)	$> 40,000$ (12)
6	$> 40,000$ **(21)**	$> 40,000$ (81)	$> 40,000$ (32)	$> 40,000$ (43)
7	$> 50,000$ **(1)**	$> 50,000$ (37)	$> 50,000$ (3)	$> 50,000$ **(1)**
8	$> 50,000$ (26)	$> 50,000$ (67)	$> 50,000$ **(21)**	$> 50,000$ (32)
9	$> 60,000$ (17)	$> 60,000$ (40)	$> 60,000$ **(11)**	$> 60,000$ (13)
10	$> 60,000$ (68)	$> 60,000$ (84)	$> 60,000$ **(50)**	$> 60,000$ (51)
		Average number of function evaluations		
1	**227**	276	289	304
2	**830**	3730	1574	1273
3	**1166**	2971	1335	1728
4	**2116**	> 8413	2439	3136
5	> 8601	$> 20,811$	> 9574	$> 13,827$
6	$> 21,246$	$> 33,617$	$> 20,366$	$> 25,270$
7	> 5610	$> 21,002$	> 7172	> 5147
8	$> 23,319$	$> 35,510$	$> 18,349$	$> 23,779$
9	$> 18,492$	$> 28,696$	$> 17,078$	$> 16,227$
10	$> 46,486$	$> 52,168$	$> 38,942$	$> 39,156$
		Median number of function evaluations		
1	**203**	218	278	273
2	**472**	658	941	637
3	**812**	1055	959	1183
4	**2116**	8412	2438	3136
5	4706	10,680	**4060**	7000
6	21,548	40,000	**15,120**	30,842
7	**2498**	5320	3648	2883
8	15,857	50,000	**9416**	17,730
9	7268	14,215	8204	**5780**
10	60,000	60,000	**58,488**	60,000
		Average Friedman ranks based on the number of function evaluations		
1	**1.73**	2.40	2.86	3.01
2	**1.69**	2.87	2.90	2.55
3	**1.96**	2.45	2.65	2.94
4	**2.05**	2.87	2.49	2.59
5	**1.98**	2.90	2.41	2.7
6	**2.07**	3.23	2.09	2.61
7	**2.11**	2.78	2.80	2.33
8	2.27	3.03	**2.21**	2.50
9	**2.30**	2.89	2.59	2.23
10	2.66	3.03	**2.16**	2.17
		Average CPU time (seconds) / Average Friedman rank based on CPU time		
1	**0.0268 /1.54**	0.0307/1.82	0.0574/3.36	0.0588/3.28
2	**0.1111/1.61**	1.6768/2.96	0.2241/2.96	0.1527/2.47
3	0.1632/**2.10**	1.1945/2.98	**0.1424**/2.35	0.1715/2.57
4	0.5462/**2.13**	5.5757/3.03	**0.2368**/2.24	0.3717/2.6
5	2.0594/2.43	12.9883/2.83	**0.8119/2.0**	1.9542/2.74
6	9.6642/2.76	27.3253/3.64	**1.6416/1.51**	2.7487/2.09
7	1.3881/2.79	7.7171/2.72	0.5855/2.31	**0.5680/2.18**
8	8.8312/2.81	11.7751/3.81	**1.4657/1.87**	2.2970/2.14
9	7.6537/2.84	12.1719/2.73	**1.8955**/2.29	2.2701/**2.14**
10	25.7386/3.28	22.487/3.34	**3.1162/1.48**	4.3864/1.9

">" indicates that the maximum number of function evaluations was reached; the number in parentheses indicates the number of failed problems.

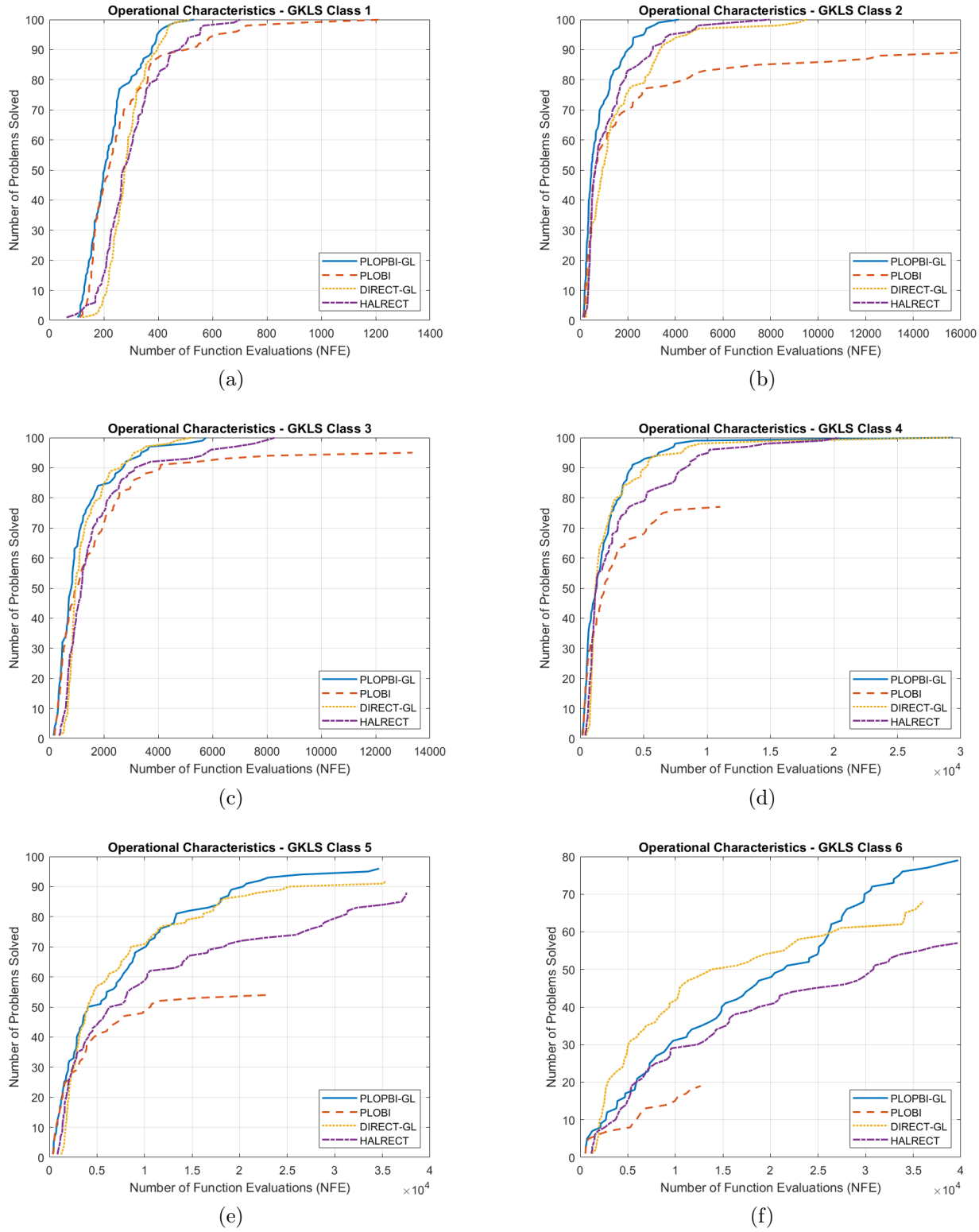


Figure 6. Number of successfully solved GKLS problems for dimensions $n = 2, 3, 4$

The CPU time results show a similar but not identical trend. The PLOPBi-GL algorithm performs well in terms of CPU time for the first several classes, where it generally requires less computational time than the other algorithms. However, its CPU-time Friedman rank is not always the best, particularly for the more difficult classes. This difference between the NFE and CPU-time rankings indicates that a lower number of function evaluations does not always lead to a lower CPU time. The higher CPU time may be related to the local search stage, particularly the use of Compromise Programming for selecting promising hyperrectangles. This additional selection process may

increase the computational time even when the number of function evaluations is relatively low.

Overall, these results indicate that the PLOPBi-GL algorithm is particularly suitable for problems where function evaluation efficiency is an important concern. For more difficult problems, the algorithm remains competitive in terms of NFE, but its higher number of failed runs and computational cost should be considered.

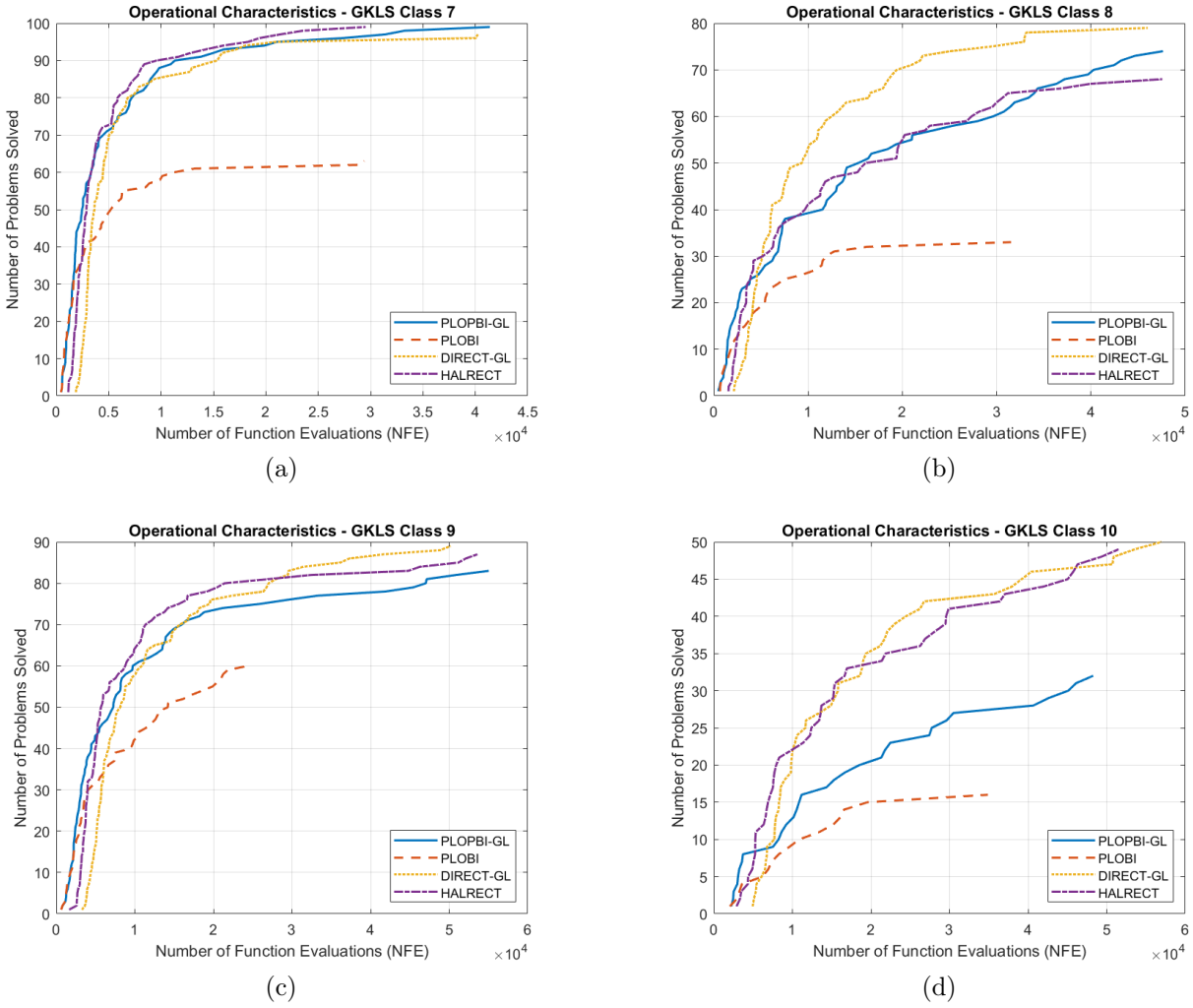


Figure 7. Number of successfully solved GKLS problems for $n = 5$ and $n = 6$

6. Conclusion

In this paper, a new strategy for selecting potentially optimal hyperrectangles within the Pareto-Lipschitzian framework has been proposed. The proposed strategy employs a two-stage Pareto approach. In the first stage, a parameterized mechanism is introduced to enhance global exploration, while in the second stage, a reduction procedure based on the Compromise Programming solution is applied to strengthen local exploitation. The resulting algorithm, called the PLOPBi-GL algorithm, also incorporates the 'divide-best' approach to support the convergence of the search process.

The effectiveness of the proposed algorithm was evaluated through numerical experiments on the Hedar and GKLS test functions. The results on the Hedar test functions show that the PLOPBi-GL algorithm is competitive with, and generally outperforms, the compared DIRECT-type algorithms in terms of the number of function evaluations over different categories of optimization problems, including both unimodal and multimodal functions. The results on the GKLS test functions also show that PLOPBi-GL generally requires fewer function evaluations than the PLOBi, DIRECT-GL, and HALRECT algorithms, particularly for the easier and medium-difficulty classes. The PLOPBi-GL algorithm also achieves a lower average Friedman rank based on the number of function evaluations across all

ten GKLS classes. However, for the more difficult GKLS classes, the number of failed runs increases, and the additional Pareto selection and Compromise Programming introduce extra computational cost in terms of CPU time. Overall, the numerical results indicate that the proposed global-local selection mechanism provides a good balance between exploration and exploitation and can reduce the number of function evaluations required to locate the global minimizer, while the additional computational cost should be considered for more difficult problems.

Acknowledgments

None.

Fundings

None.

Conflict of interest

The authors declare they have no competing interests.

Author Contributions

Conceptualization: All authors
Formal analysis: Mira Mustika
Investigation: Mira Mustika
Methodology: Mira Mustika, Salmah Salmah
Software: Mira Mustika
Supervision: Salmah Salmah, Indarsih Indarsih
Writing–original draft: Mira Mustika
Writing–review & editing: Salmah Salmah, Indarsih Indarsih

Availability of data

No new datasets were generated during the current study. The numerical experiments were conducted using publicly available benchmark functions, including the Hedar test set and the GKLS test functions. The benchmark functions and their corresponding sources are appropriately referenced in the manuscript.

AI tools statement

AI tools were used to improve the language and readability of the manuscript.

References

1. Pardalos PM, Romeijn HE, Tuy H. Recent developments and trends in global optimization. *J Comput Appl Math*. 2000;124(1-2):209-228. https://doi.org/10.1016/S0377-0427(00)00425-8
2. Alarie S, Audet C, Gheribi AE, Kokkolaras M, Le Digabel S. Two decades of blackbox optimization applications. *EURO J Comput Optim*. 2021;9:100011. https://doi.org/10.1016/j.ejco.2021.100011
3. Chaudhary R, Verma P, Salgotra R, Gandomi AH. Five decades of genetic algorithms: A systematic and bibliometric review (1975-2025). *Arch Comput Methods Eng*. 2026;1-75. https://doi.org/10.1007/s11831-025-10487-2
4. Benedetti R, Dickson MM, Espa G, Pantalone F, Piersimoni F. A simulated annealing-based algorithm for selecting balanced samples. *Comput Stat*. 2022;37(1):491-505. https://doi.org/10.1007/s00180-021-01113-3
5. Okulewicz M, Zaborski M, Mańdziuk J. Self-adapting particle swarm optimization for continuous black box optimization. *Appl Soft Comput*. 2022;131:109722. https://doi.org/10.1016/j.asoc.2022.109722
6. Shadkam E, Safari S, Abdollahzadeh SS. Finally, which meta-heuristic algorithm is the best one? *Int J Decis Sci Risk Manag*. 2021;10(1-2):32-50. https://doi.org/10.1504/IJDSRM.2021.117555
7. Liberti L, Kucherenko S. Comparison of deterministic and stochastic approaches to global optimization. *Int Trans Oper Res*. 2005;12(3):263-285. https://doi.org/10.1111/j.1475-3995.2005.00503.x
8. Vu KK, D'Ambrosio C, Hamadi Y, Liberti L. Surrogate-based methods for black-box optimization. *Int Trans Oper Res*. 2017;24(3):393-424. https://doi.org/10.1111/itor.12292
9. Liang J, Lou Y, Yu M, Bi Y, Yu K. A survey of surrogate-assisted evolutionary algorithms for expensive optimization. *J Membr Comput*. 2025;7(2):108-127. https://doi.org/10.1007/s41965-024-00165-w
10. Jones DR, Perttunen CD, Stuckman BE. Lipschitzian optimization without the Lipschitz constant. *J Optim Theory Appl*. 1993;79(1):157-181. https://doi.org/10.1007/BF00941892
11. Stripinis L, Paulavičius R. *Derivative-Free DIRECT-Type Global Optimization: Applications and Software*. Cham, Switzerland: Springer; 2023. https://doi.org/10.1007/978-3-031-46537-6
12. Li C, Chen Y, Yang X, Wang Z, Lu Z, Chi X. Intelligent Black-Litterman portfolio optimization using a decomposition-based multi-objective DIRECT algorithm. *Appl Sci*. 2022;12(14):7089. https://doi.org/10.3390/app12147089
13. Rousseau A, Pagerit S, Gao DW. Plug-in hybrid electric vehicle control strategy parameter optimization. *J Asian Electr Veh*. 2008;6(2):1125-1133. https://doi.org/10.4130/jaev.6.1125
14. Panday A, Bansal HO. Fuel efficiency optimization of input-split hybrid electric vehicle using DIRECT algorithm. In: *2014 9th International Conference on Industrial and Information Systems (ICIIS)*. IEEE; 2014:1-6. https://doi.org/10.1109/ICIINFS.2014.7036640

15. Barmuta P, Mercuri M, Soh PJ, Karsmakers P, Vandenbosch GAE, Leroux P, *et al.* Radar range improvement using gradient-free optimization for health care applications. In: *2016 21st International Conference on Microwave, Radar and Wireless Communications (MIKON).* IEEE; 2016:1-4. https://doi.org/10.1109/MIKON.2016.7491968
16. Cong S, Wang YH, Cheng JY. Coal mine microseismic velocity model inversion based on first arrival time difference. *Arab J Geosci.* 2019;12(1):5. https://doi.org/10.1007/s12517-018-4172-4
17. Larsson E, Simonsen MH, Mao W. DIRECT optimization algorithm in weather routing of ships. In: *Proceedings of the International Offshore and Polar Engineering Conference.* ISOPE; 2015:1207-1214. Accessed October 28, 2025. https://onepetro.org/ISOPEIOPEC/proceedings-abstract/ISOPE15/ISOPE15/ISOPE-I-15-170/14504
18. Hao J, Yu Z, Zhao Z, Shen P, Zhan X. Optimization of key parameters of energy management strategy for hybrid electric vehicle using DIRECT algorithm. *Energies.* 2016;9(12):997. https://doi.org/10.3390/en9120997
19. Wang N, Tsai CM, Cha KC. A study of parallel efficiency of modified DIRECT algorithm applied to thermohydrodynamic lubrication. *J Mech.* 2009;25(2):143-150. https://doi.org/10.1017/S1727719100002598
20. Zheng C, Calvin J, Gotsman C. A DIRECT-type global optimization algorithm for image registration. *J Glob Optim.* 2021;79(2):431-445. https://doi.org/10.1007/s10898-020-00914-y
21. Campana EF, Diez M, Liuzzi G, Lucidi S, Pellegrini R, Piccialli V, *et al.* A multi-objective DIRECT algorithm for ship hull optimization. *Comput Optim Appl.* 2018;71(1):53-72. https://doi.org/10.1007/s10589-017-9955-0
22. Wachowiak MP, Peters TM. High-performance medical image registration using new optimization techniques. *IEEE Trans Inf Technol Biomed.* 2006;10(2):344-353. https://doi.org/10.1109/TITB.2006.864476
23. Scitovski R, Sabo K. Application of the DIRECT algorithm to searching for an optimal k-partition of the set A $\subset \mathbb{R}^n$ and its application to the multiple circle detection problem. *J Glob Optim.* 2019;74(1):63-77. https://doi.org/10.1007/s10898-019-00743-8
24. Shen J, Dusmez S, Khaligh A. Optimization of sizing and battery cycle life in battery/ultracapacitor hybrid energy storage systems for electric vehicle applications. *IEEE Trans Ind Inform.* 2014;10(4):2112-2121. https://doi.org/10.1109/TII.2014.2334233
25. Liuzzi G, Lucidi S, Piccialli V. Exploiting derivative-free local searches in DIRECT-type algorithms for global optimization. *Comput Optim Appl.* 2016;65(2):449-475. https://doi.org/10.1007/s10589-015-9741-9
26. Liu Q, Yang G, Zhang Z, Zeng J. Improving the convergence rate of the DIRECT global optimization algorithm. *J Glob Optim.* 2017;67(4):851-872. https://doi.org/10.1007/s10898-016-0447-z
27. Paulavičius R, Sergeyev YD, Kvasov DE, Žilinskas J. Globally-biased BIRECT algorithm with local accelerators for expensive global optimization. *Expert Syst Appl.* 2020;144:113052. https://doi.org/10.1016/j.eswa.2019.113052
28. Paulavičius R, Chiter L, Žilinskas J. Global optimization based on bisection of rectangles, function values at diagonals, and a set of Lipschitz constants. *J Glob Optim.* 2018;71(1):5-20. https://doi.org/10.1007/s10898-016-0485-6
29. Sergeyev YD, Kvasov DE. Global search based on efficient diagonal partitions and a set of Lipschitz constants. *SIAM J Optim.* 2006;16(3):910-937. https://doi.org/10.1137/040621132
30. Guessoum N, Chiter L. Diagonal partitioning strategy using bisection of rectangles and a novel sampling scheme. *MENDEL.* 2023;29(2):131-146. https://doi.org/10.13164/mendel.2023.2.131
31. Stripinis L, Paulavičius R. Lipschitz-inspired HALRECT algorithm for derivative-free global optimization. *J Glob Optim.* 2024;88(1):139-169. https://doi.org/10.1007/s10898-023-01296-7
32. Jones DR. The DIRECT global optimization algorithm. In: Floudas CA, Pardalos PM, eds. *Encyclopedia of Optimization.* Boston, MA: Springer; 2001:431-440. https://doi.org/10.1007/0-306-48332-7_93
33. Mockus J. On the Pareto optimality in the context of Lipschitzian optimization. *Informatica.* 2011;22(4):521-536. https://doi.org/10.15388/Informatica.2011.340
34. Mockus J, Paulavičius R. On the reduced-set Pareto-Lipschitzian optimization. *Comput Sci Tech.* 2013;1(2):184-192. https://doi.org/10.15181/csat.v1i2.84
35. Stripinis L, Paulavičius R, Žilinskas J. Improved scheme for selection of potentially optimal hyper-rectangles in DIRECT. *Optim Lett.* 2018;12(7):1699-1712. https://doi.org/10.1007/s11590-017-1228-4
36. Mustika M, Salmah, Indarsih. A new DIRECT-type algorithm based on bisection of rectangles and diagonal sampling with the Pareto approach. *IAENG Int J Appl Math.* 2024;54(7):1352-1361. Accessed August 18, 2025. https://www.iaeng.org/IJAM/issues_v54/issue_7/IJAM_54_7_14.pdf
37. Stripinis L, Paulavičius R. DIRECTGO: A new DIRECT-type MATLAB toolbox for derivative-free global optimization. *ACM Trans Math Softw.* 2022;48(4):1-46. https://doi.org/10.1145/3559755
38. Stripinis L, Paulavičius R. An empirical study of various candidate selection and partitioning techniques in the DIRECT framework. *J Glob Optim.*

2024;88(3):723-753.
https://doi.org/10.1007/s10898-022-01185-5

39. Jones DR, Martins JRRA. The DIRECT algorithm: 25 years later. *J Glob Optim.* 2021;79(3):521-566.
https://doi.org/10.1007/s10898-020-00952-6
40. Mockus J, Paulavičius R, Rusakevičius D, Šešok D, Žilinskas J. Application of reduced-set Pareto-Lipschitzian optimization to truss optimization. *J Glob Optim.* 2017;67(1-2):425-450.
https://doi.org/10.1007/s10898-015-0364-6
41. Zeleny M. A concept of compromise solutions and the method of the displaced ideal. *Comput Oper Res.* 1974;1(3-4):479-496.
https://doi.org/10.1016/0305-0548(74)90064-1
42. Romero C, Rehman T. Chapter five: Compromise programming. In: *Multiple Criteria Analysis for Agricultural Decisions.* Amsterdam, Netherlands: Elsevier; 2003:63-78.
https://doi.org/10.1016/S0926-5589(03)80007-9
43. Hedar A. *Test Functions for Unconstrained Global Optimization* [online]. 2005. Accessed July 23, 2023. http://www-optima.amp.i.kyoto-u.ac.jp/member/student/hedar/Hedar_files/TestGO.htm.
44. Molga M, Smutnicki C. *Test Functions for Optimization Needs* [online]. Wrocław, Poland: Wrocław University of Technology; 2005. Accessed July 9, 2024. https://robertmarks.org/Classes/ENGR5358/Papers/functions.pdf.
45. Gaviano M, Kvasov DE, Lera D, Sergeyev YD. Algorithm 829: Software for generation of classes of test functions with known local and global minima for global optimization. *ACM Trans Math Softw.* 2003;29(4):469-480.
https://doi.org/10.1145/962437.962444
46. Lera D, Sergeyev YD. GOSH: Derivative-free global optimization using multi-dimensional space-filling curves. *J Glob Optim.* 2018;71(1):193-211.
https://doi.org/10.1007/s10898-017-0589-7

Mira Mustika *is a full-time lecturer at the Department of Mathematics, Faculty of Science, Institut Teknologi Sumatera, Indonesia. She is part of the Industrial and Financial Mathematics research group.*
https://orcid.org/0009-0002-2525-3752

Salmah *is a full professor in the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, in the Applied Mathematics Research Group. Her expertise is in the field of optimization, control theory, and dynamic game.*
https://orcid.org/0000-0002-5774-4321

Indarsih *is a full-time lecturer in the Department of Mathematics, Faculty of Mathematics and Natural Sciences, Universitas Gadjah Mada, in the Applied Mathematics Research Group. Her expertise is in the field of mathematical science, operations research, and applied mathematics.*
https://orcid.org/0000-0001-9065-8890

An International Journal of Optimization and Control: Theories & Applications (https://accscience.com/journal/ijocta)

This work is licensed under a Creative Commons Attribution 4.0 International License. The authors retain ownership of the copyright for their article, but they allow anyone to download, reuse, reprint, modify, distribute, and/or copy articles in IJOCTA, so long as the original authors and source are credited. To see the complete license contents, please visit http://creativecommons.org/licenses/by/4.0/.