Journal of Global Optimization (2024) 88:171–197
https://doi.org/10.1007/s10898-023-01297-6

A new partition method for DIRECT-type algorithm based on minimax design

Kai Jia[1] · Xiaojun Duan[1] · Zhengming Wang[2] · Taihe Yi[2] · Liang Yan[1] · Xuan Chen[1]

Received: 25 April 2022 / Accepted: 12 May 2023 / Published online: 23 May 2023
© The Author(s) 2023

Abstract
This article presents a new DIRECT-type SCABALL (scattering balls) algorithm with a new partition method for derivation-free optimization problems. It does not focus on dividing the region of interest into specific geometric shapes, but rather scatters several balls to cover it. In SCABALL, several potential optimal regions are selected at each iteration, and they are covered by smaller balls sequentially. In this way, the SCABALL ensures the everywhere dense convergence. The center points and radii of the scattered balls significantly influence the efficiency of SCABALL; therefore, the minimax designs are used in the initial and sequential stages to obtain better coverage. The SCABALL parameters, including the number of balls and their radii, were analyzed by numerical investigation. We provided the empirical choices for those parameters and found that the balls' radii can be contracted to balance efficiency and global convergence. Numerical experiments show that the SCABALL algorithm is locally biased and robust.

Keywords Derivative-free optimization · DIRECT-type algorithm · Minimax design · Covering radius

1 Introduction

Many problems in science and engineering can be state as derivation-free optimization problems [5, 18], such as decision-making [42], engineering design [43], molecular biology [8], system and database design [19], power generation [1], surgery [44], and astronomy [4]. In this paper, we focus on the global optimization problem in the following form:

$$\min_{\boldsymbol{x}\in\mathcal{X}} f(\boldsymbol{x}), \tag{1.1}$$

where $\mathcal{X} = [\boldsymbol{a}, \boldsymbol{b}] = \{\boldsymbol{x} \in \mathbb{R}^d : a_i \le x_i \le b_i, i = 1, 2, \ldots, d\}$ is the region of interest and $f : \mathbb{R}^d \to \mathbb{R}$ is a deterministic function whose derivatives are neither symbolically nor

✉ Xiaojun Duan
xjduan@nudt.edu.cn

[1] College of Science, National University of Defense Technology, Changsha 410005, Hunan, China

[2] College of Systems Engineering, National University of Defense Technology, Changsha 410005, Hunan, China

numerically available. There are two classes of algorithms to solve problem (1.1): the statistic heuristic algorithms, such as genetic algorithm [11], simulated annealing algorithm [17], and particle swarm algorithm [16]; the deterministic algorithms, such as partition-based method [14] and space-filling-curve-based methods [35]. In general, the statistic heuristic algorithms are flexible, but the quality of solution cannot be guaranteed. By contrast, the deterministic algorithms can provide general methods to obtain a global or approximately global optimum.

[15] and [13] proposed a well-known deterministic algorithm, called DIRECT (dividing rectangles). The DIRECT algorithm partitions $\mathcal{X}$ into several hyper-rectangles $\mathcal{R}_i$, and evaluates $f(\boldsymbol{x}_i)$, where $\boldsymbol{x}_i$ is the center of $\mathcal{R}_i$. In each iteration, DIRECT selects some potential optimal regions (POR) and partitions them into smaller ones. The simplicity and efficiency of the DIRECT algorithm attracted considerable interest from the optimization community [39]. Since the original algorithm was presented, many scholars have modified or extended DIRECT in various ways. In this paper, we summarize only part of these DIRECT-type algorithms in the following 3 aspects, and a more detailed review can be found in the recent article [14]. First, some studies have focused on different methods for partitioning regions [20, 29, 31, 34]. The partition methods are the focus of this paper, and will be address them in detail in Sect. 2. Second, some articles have focused on modifying the POR selection scheme [9, 21, 22, 24, 25, 30, 32, 34, 41]. They set the algorithm to search more locally or globally or switch the search state regularly by redefining the set of POR. Third, hybrid methods have been proposed to accelerate the DIRECT convergence [20, 23, 32]. These researchers combined DIRECT with local optimizer or meta-model technique to overcome DIRECT's low efficiency when faced with high dimensionality or accuracy.

In this paper, we propose the SCABALL (scattering balls) algorithm, a new DIRECT-type algorithm with a novel partition method. The remainder of this article is organized as follows. Section 2 summarizes the schemes of DIRECT and DIRECT-type algorithms with different partitions. Then we explain the idea, process, and property of SCABALL in Sect. 3. In Sect. 4, we elaborate on the implementation and contraction of partitions in SCABALL. Section 5 displays some numerical experiments, including parameter tuning and comparison results. Finally, the main conclusions and further discussions are in Sect. 6.

2 Scheme of DIRECT-type algorithm

We summarize the main procedure for the DIRECT-type algorithms in Fig. 1. The *Partition*, *Sample* and *Evaluate* steps may be mixed up in some articles, we separate them to clearly compare different algorithms. It is worth noting that the *Select* and *Partition* of PORs depend on the *Sample* and *Evaluate* steps in most DIRECT-type algorithms, thus the first sampling and evaluation are included in the *Initialization*.

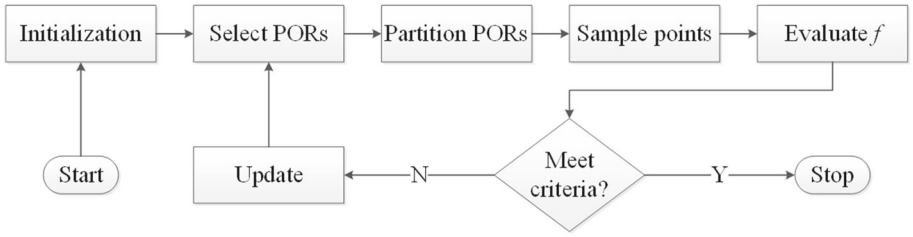


Fig. 1 Flow chart of DIRECT-type algorithms

The key idea of DIRECT is the selection of POR, which is based on $f(\mathcal{R}_i)$ and $d(\mathcal{R}_i)$, where

$$f(\mathcal{R}_i) \triangleq f(\boldsymbol{x}_i), \quad d(\mathcal{R}_i) \triangleq \sup_{\boldsymbol{x}\in\mathcal{R}_i} \|\boldsymbol{x}_i - \boldsymbol{x}\|, \tag{2.1}$$

$\|\cdot\|$ denotes the Euclidean norm. Assume f is Lipschitz continuous:

$$|f(\boldsymbol{x}_1) - f(\boldsymbol{x}_2)| \le K\|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \quad \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathcal{X}, \tag{2.2}$$

where $K \ge 0$ is the Lipschitz constant. Then we get a lower bound of f in $\mathcal{R}_i$:

$$f(\boldsymbol{x}) \ge f(\mathcal{R}_i) - K \cdot d(\mathcal{R}_i), \quad \forall \boldsymbol{x} \in \mathcal{R}_i. \tag{2.3}$$

Since we cannot use the derivatives of f, the Lipschitz constant K is assumed to be unknown. Define the POR as those $\mathcal{R}_i$ whose lower bound is minimized under some $\hat{K}$. That is to say, $\mathcal{R}_i$ is a POR if there exists a constant $\hat{K}$ so that

$$f(\mathcal{R}_i) - \hat{K} \cdot d(\mathcal{R}_i) \le f(\mathcal{R}_j) - \hat{K} \cdot d(\mathcal{R}_j), \quad \forall j \ne i. \tag{2.4}$$

To avoid the algorithm focusing on a trivial solution, [15] added another condition:

$$f(\mathcal{R}_i) - \hat{K} \cdot d(\mathcal{R}_i) \le f_{\min} - \epsilon|f_{\min}|, \tag{2.5}$$

where $f_{\min}$ is the minimum evaluation so far and ϵ is a balance parameter. The benefit of Eq. (2.1) is that both $d(\mathcal{R}_i)$ and $f(\boldsymbol{x}_i)$ can be taken as the properties of $\mathcal{R}_i$. Then every $\mathcal{R}_i$ can be represented by a dot in two-dimensional diagram (see Fig. 2). The horizontal and vertical axes represent $d(\mathcal{R})$ and $f(\mathcal{R})$, respectively. Note that there is an extra dot at $(0, f_{\min} - \epsilon|f_{\min}|)$ that does not represent any region but is determined by condition (2.5). The potential PORs satisfying conditions (2.4) and (2.5) are those on the lower right area of the convex hull of the dot cloud. The upper right $\mathcal{R}_i$ have larger size, which tends towards global exploration, while the lower left $\mathcal{R}_i$ have better responses, which tends towards local exploitation. This scheme is an effective trade-off between global exploration and local exploitation. Notably the $\mathcal{R}_i$ with the best evaluation is always selected [9]. As the iterations approach to infinity, the $\max_i d(\mathcal{R}_i)$ must approach zero. As a result, the points sampled by DIRECT will be "everywhere dense" [14].

In this paper, we focus on the following five variants with different partitions: the revised DIRECT [13], adaptive diagonal curves (ADC) [34], dividing simplices at vertices (DISIMPL-V), dividing simplices at centers (DISIMPL-C) [29, 30], bisecting rectangles (BIRECT) [31, 32]. DIRECT and ADC both divide the region of interest into three hyper-rectangles, then DIRECT evaluates f at the center of the hyper-rectangles, while ADC does it at two vertices of the main diagonal. DISIMPL-V and DISIMPL-C divide the region of interest into simplices, then evaluate f at the vertices and centroid of each simplex respectively. BIRECT divides the region of interest into two rectangles, then evaluates f at the diagonal trines. Although the geometric shapes of $\mathcal{R}_i$ in DIRECT-type algorithms are different, one thing they have in common is that they all select PORs according to $f(\mathcal{R}_i)$ and $d(\mathcal{R}_i)$. We summarize the definitions of $f(\mathcal{R}_i)$ and $d(\mathcal{R}_i)$ as follows. For the definition of $f(\mathcal{R}_i)$, DIRECT and DISIMPL-C allocate only one point $\boldsymbol{x}_i$ in each $\mathcal{R}_i$; therefore, $f(\mathcal{R}_i) = f(\boldsymbol{x}_i)$. Conversely, there are multiple points in each $\mathcal{R}_i$ when either ADC, DISIMPL-V, or BIRECT is used. ADC defines $f(\mathcal{R}_i) = [f(\boldsymbol{v}_{i1}) + f(\boldsymbol{v}_{i2})]/2$ where $\boldsymbol{v}_{i1}$ and $\boldsymbol{v}_{i2}$ are the vertices of the main diagonal of $\mathcal{R}_i$. DISIMPL-V and BIRECT define $f(\mathcal{R}_i) = \min_{\boldsymbol{x}_j\in\mathcal{R}_i} f(\boldsymbol{x}_j)$ to obtain the lower bound of f. For the definition of $d(\mathcal{R}_i)$, DIRECT and DISIMPL-C share a similar definition as Eq. (2.1). ADC defines $d(\mathcal{R}_i) = \|\boldsymbol{v}_{i1} - \boldsymbol{v}_{i2}\|/2$, which is virtually identical to DIRECT and DIRECT. BIRECT defines $d(\mathcal{R}_i) = 2\|\boldsymbol{v}_{i1} - \boldsymbol{v}_{i2}\|/3$, and DISIMPL-V defines

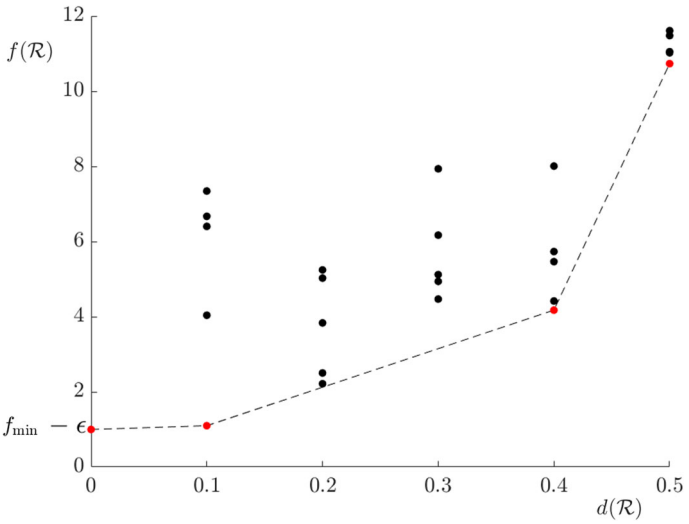


Fig. 2 Identifying potentially optimal hyper-rectangles in DIRECT

$d(\mathcal{R}_i)$ as the largest side length of $\mathcal{R}_i$. All these DIRECT-type algorithms follow the scheme in Fig. 1. For clarity, the main steps of these algorithms are included in columns 2–6 of Table 1. The numbers in parentheses in the *Initialize* row represent the quantity of initial evaluations, and the numbers in parentheses in the *Evaluate* row represent the quantity of evaluations in each POR. The demonstrations of the 4-th iteration of DIRECT-type algorithms are shown in Fig. 3a–e, where the contour corresponds to the Branin function [38]. The regions bordered in red represent the selected PORs, the red dots denote the new sampled points, the red dash lines display the partitions of sub-regions. In this article, we focus on the influence of different partitions on the algorithm. Therefore, we selected the PORs uniformly using Eq. (2.4) and (2.5) in the above algorithms.

3 SCABALL algorithm

Let us start with the basic idea of SCABALL. Note that all existing partition methods introduced in Sect. 2 follow the two principles [34]:

$$\mathcal{X} = \cup_{i=1}^{n} \mathcal{R}_i, \quad \mathcal{R}_i \cap \mathcal{R}_j = \partial\mathcal{R}_i \cap \partial\mathcal{R}_j, \ i \neq j, \tag{3.1}$$

where ∂ denotes the boundary. Namely, all $\mathcal{R}_i$ form $\mathcal{X}$ exactly, and different $\mathcal{R}_i$ intersect only at the boundary. SCABALL is a new partition method based on the Voronoi tessellation of sampled points. The Voronoi tessellation divides $\mathcal{X}$ into sub-regions $\mathcal{R}_i$, where $\mathcal{R}_i$ is consisting of points closer to $\boldsymbol{x}_i$ than to any other samples. The left of Fig. 4 demonstrates the Voronoi tessellation of 5 random samples. Clearly, Voronoi tessellation follows the principles (3.1). The usage of the Voronoi tessellation of the search domain is not a new idea: this technique is often used in interpolation [2], simulation-based optimization [20], and linear constrained optimization [3]. However, due to the high computational costs involved, such methods could be suitable for low-dimensional problems only. To handle this problem, we first note that the selection of POR is not related to the shape of $\mathcal{R}_i$ but only to $d(\mathcal{R}_i)$, then

Table 1 Main steps of DIRECT-type algorithms

Algorithm	DIRECT	ADC	DISIMPL-V	DISIMPL-C	BIRECT	SCABALL
Initialize	Centroid of rectangle (1)	Vertices of diagonal (2)	Vertices of simplices (2^s)	Centroids of simplices ($s!$)	Trines of diagonal (2)	Initial mM design (n_{ini})
$\mathcal{R}_i$	Rectangles	Rectangles	Simplices	Simplices	Rectangles	Balls
$f(\mathcal{R}_i)$	$f(\boldsymbol{x}_i)$	$\frac{f(\boldsymbol{v}_{i1})+f(\boldsymbol{v}_{i2})}{2}$	$\min_{\boldsymbol{x}_j \in \mathcal{R}_i} f(\boldsymbol{x}_j)$	$f(\boldsymbol{x}_i)$	$\min_{\boldsymbol{x}_j \in \mathcal{R}_i} f(\boldsymbol{x}_j)$	$f(\boldsymbol{x}_i)$
$d(\mathcal{R}_i)$	Half of the diagonal	Half of the diagonal	The largest side length	Centroid to the farthest vertex	Two-thirds of the diagonal	Related to mM distance
Select	The selection of POR is consistent with DIRECT based on $f(\mathcal{R}_i)$, $d(\mathcal{R}_i)$ and ϵ					
Partition	Trisect along largest side	Trisect along largest side	Bisect along largest side	Trisect along largest side	Bisect along largest side	Scatter balls around
Sample	Centroid of new rectangles	Vertices of new diagonal	Vertices of new simplices	Centroid of new simplices	Trines of new diagonals	Sequential mM design
Evaluate	All sampled points (2)	Non-repeated points (≤ 2)	Non-repeated points (≤ 1)	All sampled points (2)	Non-repeated points (2)	Non-repeated points ($n_{\text{seq}} - 1$)

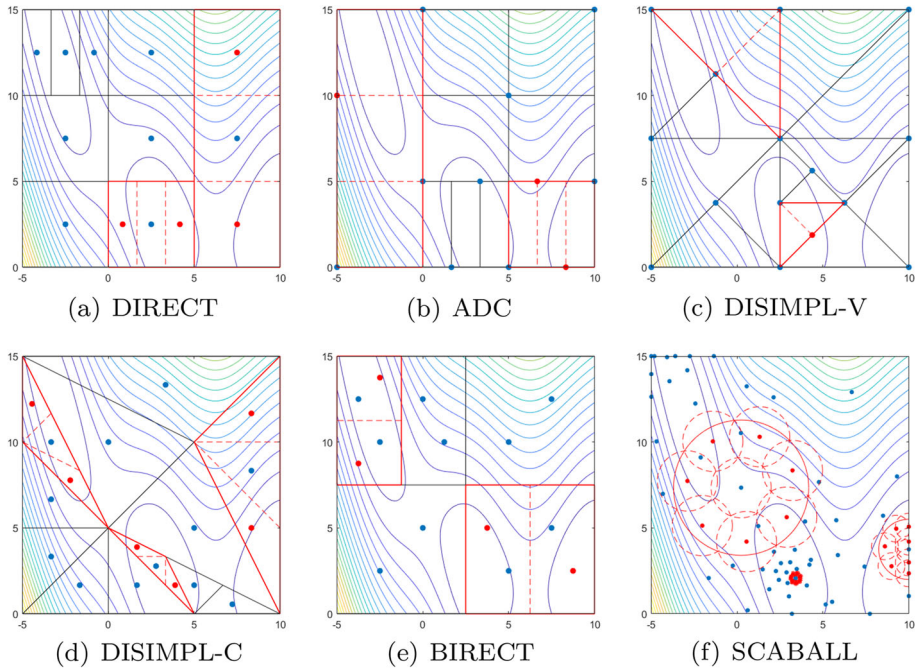


Fig. 3 Demonstrations of different DIRECT-type algorithms at 4-th iteration

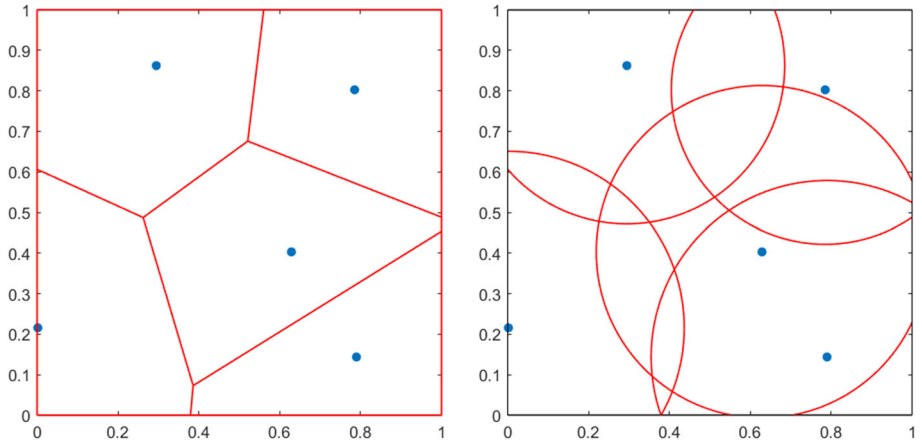


Fig. 4 Examples of the Voronoi tessellation and ball coverage

we relax the principles (3.1) as

$$\mathcal{X} \subseteq \cup_{i=1}^{n} \mathcal{R}_i . \tag{3.2}$$

The proposed SCABALL algorithm does not concentrate on dividing $\mathcal{X}$ into specific geometry, but rather scatters several balls to cover the region of interest, see the right of Fig. 4. This reduces the calculation and make it possible to solve problems with higher dimension

(relatively). The complete description of the SCABALL is shown in Algorithm 1. The rest of this section will elaborate the process of SCABALL and its convergence.

Algorithm 1: SCABALL

Input: $f, \mathcal{X}, \epsilon, n_{\text{ini}}, n_{\text{seq}}, \gamma$

Preparation: construct or read $X^{\text{ini}}_{n_{\text{ini}}}$ and $X^{\text{seq}}_{n_{\text{seq}}}$, calculate $d^{\text{ini}}_{\text{mM}}$ and $d^{\text{seq}}_{\text{mM}}$;

Initialization:

Normalize $\mathcal{X}$ to the unit hyper-cube $\bar{\mathcal{X}}$;

Initialize $k \leftarrow 0$, $\mathcal{R}^{(0)}_0 \leftarrow \bar{\mathcal{X}}$, $S^{(0)}_0 \leftarrow X^{\text{ini}}_{n_{\text{ini}}}$, and $I^{(0)}_0 \leftarrow \{1, 2, \ldots, n_{\text{ini}}\}$;

Evaluate f at all $\boldsymbol{x}_i \in S^{(0)}_0$, and $\mathcal{R}^{(1)}_i = \mathcal{B}(\boldsymbol{x}_i, d^{\text{ini}}_{\text{mM}}), \forall i \in I^{(0)}_0$;

$f(\mathcal{R}^{(1)}_i) \leftarrow f(\boldsymbol{x}_i), d(\mathcal{R}^{(1)}_i) \leftarrow d^{\text{ini}}_{\text{mM}}, \forall i \in I^{(0)}_0$, find $f_{\min}$, and $\boldsymbol{x}_{\min}$;

while *stopping criteria not met* **do**

  $k \leftarrow k + 1$;

  Select POR based on $f(\mathcal{R}^{(k)}_i), d(\mathcal{R}^{(k)}_i)$, and ϵ ; `// equations (2.4) and (2.5)`

  for *each* $j^* \in POR$ **do**

    $S^{(k)}_{j^*} \leftarrow T(X^{\text{seq}}_{n_{\text{seq}}}; \boldsymbol{a}^{(k)}_{j^*}, \boldsymbol{b}^{(k)}_{j^*})$; `// equations (3.11) and (3.12)`

    Evaluate f at new $\boldsymbol{x}_i \in S^{(k)}_{j^*}$, and $\mathcal{R}^{(k+1)}_i = \mathcal{B}(\boldsymbol{x}_i, r^{(k+1)}_i), \forall i \in I^{(k)}_{j^*}$;

    Update $f(\mathcal{R}^{(k+1)}_i), d(\mathcal{R}^{(k+1)}_i)$, $f_{\min}$, and $\boldsymbol{x}_{\min}$; `// equation (4.6)`

  end

end

Output: $f_{\min}$ and $\boldsymbol{x}_{\min}$

3.1 Overall scheme

As a DIRECT-type algorithm, SCABALL follows the scheme summarized in Fig. 1. To clarify the process of iteration, we denote the i-th region at k-th iteration by $\mathcal{R}^{(k)}_i$, the set of points sampled in $\mathcal{R}^{(k)}_i$ by $S^{(k)}_i$, and the index of points in $S^{(k)}_i$ by $I^{(k)}_i$.

In initialization, we first normalize $\mathcal{X}$ to a unit hyper-cube $\bar{\mathcal{X}} \triangleq [\mathbf{0}, \mathbf{1}]$. Let $\mathcal{R}^{(0)}_0 \triangleq \bar{\mathcal{X}}$ and $S^{(0)}_0$ denote the sampled points in initialization, and $\mathcal{B}(\boldsymbol{x}, r)$ denote the closed ball with center $\boldsymbol{x}$ and radius r. According to the principle (3.2),

$$\mathcal{R}^{(0)}_0 \triangleq \bar{\mathcal{X}} \subseteq \bigcup_{\boldsymbol{x}_i \in S^{(0)}_0} \mathcal{B}(\boldsymbol{x}_i, r^{(1)}_i) \triangleq \bigcup_{i \in I^{(0)}_0} \mathcal{R}^{(1)}_i. \tag{3.3}$$

We evaluate $f(\boldsymbol{x}_i)$, and define

$$f(\mathcal{R}^{(k)}_i) \triangleq f(\boldsymbol{x}_i), \quad d(\mathcal{R}^{(k)}_i) \triangleq \sup_{\boldsymbol{x} \in \mathcal{R}^{(k)}_i} \|\boldsymbol{x}_i - \boldsymbol{x}\| = r^{(k)}_i. \tag{3.4}$$

With the definition (3.4), the selection of POR can be carried out according to equations (2.4) and (2.5). Assume $\mathcal{R}^{(k)}_{j^*}$ is one of the POR selected at k-th iteration. In the partition step, several balls are scattered to cover $\mathcal{R}^{(k)}_{j^*}$ iteratively, i.e.

$$\mathcal{R}^{(k)}_{j^*} \subseteq \bigcup_{\boldsymbol{x}_i \in S^{(k)}_{j^*}} \mathcal{B}(\boldsymbol{x}_i, r^{(k+1)}_i) \triangleq \bigcup_{i \in I^{(k)}_{j^*}} \mathcal{R}^{(k+1)}_i. \tag{3.5}$$

Since $\boldsymbol{x}_{j^*}$ already exists in $\mathcal{R}_{j^*}^{(k)}$, we set $\boldsymbol{x}_{j^*} \in S_{j^*}^{(k)}$ to obtain better coverage. Although other $\boldsymbol{x}_j$ may exist in $\mathcal{R}_{j^*}^{(k)}$, we ignore it for simplicity. Then we sample $S_{j^*}^{(k)}$ in $\mathcal{R}_{j^*}^{(k)}$, evaluate $f(\boldsymbol{x}_i)$ at all points in $S_{j^*}^{(k)}$ except $\boldsymbol{x}_{j^*}$, and update $f(\mathcal{R}_i^{(k+1)}) = f(\boldsymbol{x}_i),\ d(\mathcal{R}_i^{(k+1)}) = r_i^{(k+1)},\ \forall i \in \boldsymbol{I}_{j^*}^{(k)}$. For those $\mathcal{R}_j^{(k)}$ that are not POR, we set

$$\mathcal{R}_j^{(k+1)} = \mathcal{R}_j^{(k)},\ S_j^{(k)} = \{\boldsymbol{x}_j\},\ d(\mathcal{R}_j^{(k+1)}) = d(\mathcal{R}_j^{(k)}). \tag{3.6}$$

The overall scheme of SCABALL is complete. In order to compare with other algorithms, the main steps of the SCABALL are tabulated in the last column of Table 1, and the demonstration of SCABALL is shown in Fig. 3f.

3.2 Partitions in initialization and iterations

The most important part of SCABALL is the partition denoted by Eqs. (3.3) and (3.5). The $\boldsymbol{x}_i$ and r_i significantly influence the efficiency of the SCABALL algorithm. Thus, they are not arbitrary and need delicate design. To explain this, the lower bound of f in SCABALL can also be expressed as

$$f(\boldsymbol{x}) \geq f(\mathcal{R}_i) - K \cdot d(\mathcal{R}_i), \quad \forall \boldsymbol{x} \in \mathcal{R}_i, \tag{3.7}$$

which is consistent with Eq. (2.3). In order to get a larger, which is more favorable, lower bound of f in $\mathcal{R}_i$, a smaller $d(\mathcal{R}_i)$ is preferred. To achieve this, we introduced the minimax (mM) design.

Let $X_n = \{\boldsymbol{x}_i, i = 1, 2, \ldots, n\}$ be a subset with n points on a convex region $\mathcal{R} \subseteq \bar{\mathcal{X}}$, which represents a design on $\mathcal{R}$. The mM design is a well-known space-filling design proposed by [12]. The mM-distance criterion of a design X_n in $\mathcal{R}$ is defined as follows:

$$d_{\mathrm{mM}}(X_n) \triangleq d_{\mathrm{mM}}(X_n, \mathcal{R}) = \max_{\boldsymbol{x} \in \mathcal{R}} d(\boldsymbol{x}, X_n) = \max_{\boldsymbol{x} \in \mathcal{R}} \min_{i=1,\ldots,n} \|\boldsymbol{x} - \boldsymbol{x}_i\|. \tag{3.8}$$

$d_{\mathrm{mM}}(X_n)$ corresponds to the Hausdorff distance between X_n and $\mathcal{R}$, and is also called the dispersion of X_n [27]. $d_{\mathrm{mM}}(X_n)$ can also be described as:

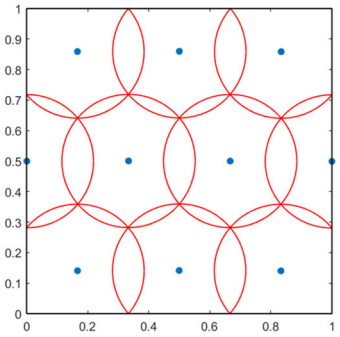
$$d_{\mathrm{mM}}(X_n) = \inf\left\{r \geq 0 \mid \mathcal{R} \subseteq \cup_{i=1}^{n} \mathcal{B}(\boldsymbol{x_i}, r)\right\}. \tag{3.9}$$

This means that the balls centered on $\boldsymbol{x_i}$ with same radius $d_{\mathrm{mM}}(X_n)$ cover $\mathcal{R}$ exactly. X_n^* is called the mM design on $\mathcal{R}$ if

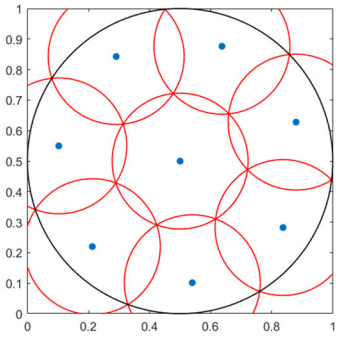
$$d_{\mathrm{mM}}(X_n^*) = \min_{X_n \in \mathcal{R}^n} d_{\mathrm{mM}}(X_n). \tag{3.10}$$

Namely, the mM design covers the region of interest with equal and minimum radii, which is a suitable partition for SCABALL.

Note that there are two differences between partitions (3.3) and (3.5): one is the geometry, and the other is the structure. First, $\mathcal{R}_1^{(0)}$ is a hyper-cube, but all $\mathcal{R}_{j^*}^{(k)}$ are hyper-balls. Second, $S_0^{(0)}$ is completely free in $\mathcal{R}_1^{(0)}$, but $S_{j^*}^{(k)}$ should take into account the relationship with $\boldsymbol{x}_{j^*}$. Because of these differences, we need two kinds of mM design. One is the initial design $X_{n_{\mathrm{ini}}}^{\mathrm{ini}}$, which is freely designed on $\mathcal{R}^{\mathrm{ini}} \triangleq \bar{\mathcal{X}}$. The other is the sequential design $X_{n_{\mathrm{seq}}}^{\mathrm{seq}}$, which is designed on $\mathcal{R}^{\mathrm{seq}} \triangleq \mathcal{B}(\mathbf{1}/2, 1/2) \subset \bar{\mathcal{X}}$ and with one point fixed at $\mathbf{1}/2$. The n_{ini} and n_{seq} on the subscript indicate the quantity of points in the design, and the *ini* and *seq* on the superscript indicate the geometry and structure of the design. Figure 5 demonstrates the examples of X_{10}^{ini} and X_8^{seq} in 2 dimensions. The construction of mM design will be detailed in Sect. 4.1, we next explain how to sample and update by mM design.



(a) X_{10}^{ini} in $[0,1]^2$ with $d_{\text{mM}}^{\text{ini}} = 0.2182$



(b) X_8^{seq} in $\mathcal{B}((0.5,0.5),0.5)$ with $d_{\text{mM}}^{\text{seq}} = 0.2225$

Fig. 5 Examples of the mM design in 2 dimensions

3.3 Sampling and updating

For the convenience, we define a linear mapping $T : [\mathbf{0}, \mathbf{1}] \mapsto [\boldsymbol{a}, \boldsymbol{b}]$ as follows:

$$T(\boldsymbol{x}) \triangleq T(\boldsymbol{x}; \boldsymbol{a}, \boldsymbol{b}) = \boldsymbol{a} + \boldsymbol{x} \circ (\boldsymbol{b} - \boldsymbol{a}),$$

where $\circ$ denotes the Hadamard product and $T(X_n) \triangleq \{T(\boldsymbol{x}_1), T(\boldsymbol{x}_2), \ldots, T(\boldsymbol{x}_n)\}$. We only design $X_{n_{\text{ini}}}^{\text{ini}}$ and $X_{n_{\text{seq}}}^{\text{seq}}$ once beforehand, then all sample points $S_i^{(k)}$ can be obtained by T with proper parameters. Since the mM designs are response-free, the constructed mM designs can be read in preparation (see Algorithm 1 Line 1). Note that $\mathcal{R}_1^{(0)} = \bar{\mathcal{X}}$, we get samples as follows:

$$S_0^{(0)} = X_{n_{\text{ini}}}^{\text{ini}}, \quad S_{j^*}^{(k)} = T(X_{n_{\text{seq}}}^{\text{seq}}; \boldsymbol{a}_{j^*}^{(k)}, \boldsymbol{b}_{j^*}^{(k)}), \tag{3.11}$$

where

$$\boldsymbol{a}_{j^*}^{(k)} = \boldsymbol{x}_{j^*}^{(k)} - \mathbf{1} \cdot d(\mathcal{R}_{j^*}^{(k)})/2, \quad \boldsymbol{b}_{j^*}^{(k)} = \boldsymbol{x}_{j^*}^{(k)} + \mathbf{1} \cdot d(\mathcal{R}_{j^*}^{(k)})/2. \tag{3.12}$$

Further more, $d(\mathcal{R}_i^{(k)})$ can be easily updated due to the property of d_{mM}. According to Eq. (3.9), all sub-regions in mM design have the same radius d_{mM}. Since $\bar{\mathcal{X}} = [\mathbf{0}, \mathbf{1}]$, d_{mM} can also be viewed as half the size ratio of sub-regions to original regions. Let $d_{\text{mM}}^{\text{ini}} \triangleq d_{\text{mM}}(X_{n_{\text{ini}}}^{\text{ini}})$ and $d_{\text{mM}}^{\text{seq}} \triangleq d_{\text{mM}}(X_{n_{\text{seq}}}^{\text{seq}})$, we have

$$\begin{aligned} d(\mathcal{R}_i^{(1)}) = r_i^{(1)} &= d_{\text{mM}}^{\text{ini}}, \ \forall i \in I_0^{(0)}, \\ d(\mathcal{R}_i^{(k+1)}) = r_i^{(k+1)} &= 2d_{\text{mM}}^{\text{seq}} \cdot d(\mathcal{R}_{j^*}^{(k)}), \ \forall i \in I_{j^*}^{(k)}. \end{aligned} \tag{3.13}$$

In this way, for any region at k-th iteration, $d(\mathcal{R}_i^{(k)})$ has the form of

$$d(\mathcal{R}_i^{(k)}) = d_{\text{mM}}^{\text{ini}}(2d_{\text{mM}}^{\text{seq}})^{k^-}, \tag{3.14}$$

where $k^- \leq k$ indicates the number of times that $\mathcal{R}_i^{(k)}$ was partitioned.

The initialization and first two iterations of SCABALL are illustrated in Fig. 6. The red solid lines denote the selected PORs in current stage, which is hyper-cube in initialization and hyper-ball in iterations. The red dash lines denote the sub-regions partitioned by SCABALL. The red dots denote the new sampled points in POR. It is worth noting that there could be

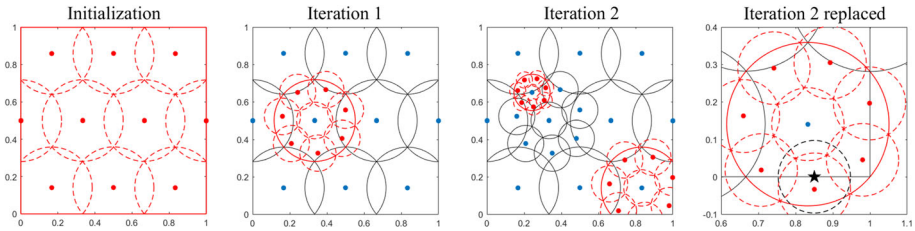


Fig. 6 Illustration of the partition method of SCABALL in 2 dimensions

some sampled points $\boldsymbol{x}_i \notin \bar{\mathcal{X}}$, see the lower right corner of *Iteration 2* in Fig. 6. In this case, we choose $\boldsymbol{x}_i' = \arg\min_{\boldsymbol{x}\in\bar{\mathcal{X}}} \|\boldsymbol{x}_i - \boldsymbol{x}\|$ to replace $\boldsymbol{x}_i$. This replacement is detailed in the last graph of Fig. 6, the black star point and black dash line denote $\boldsymbol{x}_i'$ and $\mathcal{R}_i'$ respectively, where $\mathcal{R}_i' = \mathcal{B}(\boldsymbol{x}_i', r_i)$. It is easy to prove that $\mathcal{R}_i \cap \bar{\mathcal{X}} \subseteq \mathcal{R}_i' \cap \bar{\mathcal{X}}$, if $\bar{\mathcal{X}}$ is convex and closed. Thus, this replacement does not break the coverage in Eq. (3.18), which is important to the following proof of convergence.

3.4 Global convergence

Let $S^{(0)} \triangleq S_0^{(0)}$, $I^{(k)}$ denotes the index of the points in $S^{(k)}$, where $S^{(k)} = \bigcup_{i\in I^{(k-1)}} S_i^{(k)}$ denotes the set of all points sampled by SCABALL after k iterations. Clearly, $I^{(k)} = \bigcup_{i\in I^{(k-1)}} I_i^{(k)}$. Since the selection of POR is similar to DIRECT, SCABALL always selects the largest region with the best evaluation. This guarantees the density of $S^{(k)}$, and further guarantees the global convergence of SCABALL under the assumption of continuity of f. We formally state this property as a theorem.

Theorem 3.1 *If* $d_{\mathrm{mM}}^{\mathrm{seq}} < 1/2$, *then* $\forall \boldsymbol{x} \in \bar{\mathcal{X}}$

$$d(\boldsymbol{x}, S^{(k)}) = \min_{\boldsymbol{x}_i \in S^{(k)}} d(\boldsymbol{x}, \boldsymbol{x}_i) \to 0, \quad as \quad k \to \infty.$$

Proof According to Eq. (3.3)

$$\bar{\mathcal{X}} \subseteq \bigcup_{i\in I^{(0)}} \mathcal{R}_i^{(1)}. \tag{3.15}$$

From Eqs. (3.5) and (3.6), we have

$$\mathcal{R}_j^{(k)} \subseteq \bigcup_{i\in I_j^{(k)}} \mathcal{R}_i^{(k+1)}, \ \forall j \in I^{(k-1)}, \tag{3.16}$$

which indicates that

$$\bigcup_{j\in I^{(k-1)}} \mathcal{R}_j^{(k)} \subseteq \bigcup_{i\in I^{(k)}} \mathcal{R}_i^{(k+1)}. \tag{3.17}$$

By recursion, we obtain

$$\bar{\mathcal{X}} \subseteq \bigcup_{i\in I^{(k-1)}} \mathcal{R}_i^{(k)}, \ \forall k \geq 1. \tag{3.18}$$

Let $d^{(k)} = \max_{i \in I^{(k-1)}} d(\mathcal{R}_i^{(k)})$ denote the largest radius of $\mathcal{R}_i^{(k)}$ at the k-th iteration, then

$$d(\boldsymbol{x}, S^{(k)}) \leq d^{(k)}, \ \forall \boldsymbol{x} \in \bar{\mathcal{X}}, k \geq 1. \tag{3.19}$$

Let $I_{\max}^{(k)} = \{i | d(\mathcal{R}_i^{(k)}) = d^{(k)}\}$ denote the index of the largest region at k-th iteration, and $I_*^{(k)} = \{i \in I_{\max}^{(k)} | \arg\min_i f(\mathcal{R}_i^{(k)})\}$. For those $\mathcal{R}_i^{(k)}, i \in I_*^{(k)}$, there is always $\hat{K}$ large enough that conditions (2.4) and (2.5) are satisfied, which means they are POR at the k-th iteration. According to Eq. (3.13), the POR will be partitioned into smaller regions at a rate of $2d_{\mathrm{mM}}^{\mathrm{seq}} < 1$. Since $I_{\max}^{(k)}$ is finite, $d^{(k)} \to 0$ as $k \to \infty$, which completes the proof. □

For the sequential design $X_1^{\mathrm{seq}} = \{\mathbf{1}/2\}$ on $\mathcal{B}(\mathbf{1}/2, 1/2)$, we have $d_{\mathrm{mM}}(X_1^{\mathrm{seq}}) = 1/2$. Since $d_{\mathrm{mM}}(\cdot)$ is a non-decreasing set function, $d_{\mathrm{mM}}(X_{n_{\mathrm{seq}}}^{\mathrm{seq}}) \leq 1/2$. Therefore, the condition $d_{\mathrm{mM}}^{\mathrm{seq}} < 1/2$ in Theorem 3.1 is not difficult to achieve if n_{seq} is large enough.

4 Implementation and contraction of partition

4.1 Construction of mM design

The theoretical mM design X_n^* in Eq. (3.10) is extremely difficult to construct, because evaluating $d_{\mathrm{mM}}(X_n)$ in Eq. (3.8) requires maximizing $d(\boldsymbol{x}, X_n)$ with respect to $\boldsymbol{x} \in \mathcal{R}$. There is a geometric method based on Voronoi tessellation to evaluate $d_{\mathrm{mM}}(X_n)$ [6, 7], and a corresponding algorithm to obtain X_n^* [33]. The mM designs in Fig. 5 were constructed by the geometric algorithm. While this kind of method requires calculating the Voronoi tessellation and its Chebyshev center at each iteration, which is computationally expensive and difficult to apply to the situation when $\mathcal{R}$ is a hyper-ball. In this article, we introduce a fast, approximate, and random method to obtain $X_{n_{\mathrm{ini}}}^{\mathrm{ini}}$ and $X_{n_{\mathrm{seq}}}^{\mathrm{seq}}$.

A fully-sequential space-filling design in [37] was constructed iteratively by greedily maximizing $D_\beta(\boldsymbol{x}, X_k)$, where $D_\beta(\boldsymbol{x}, X_k)$ is defined by

$$D_\beta(\boldsymbol{x}, X_k) \triangleq \min\{d(\boldsymbol{x}, X_k), \beta \cdot d(\boldsymbol{x}, \partial\mathcal{R})\}, \quad \boldsymbol{x} \in \mathcal{R}. \tag{4.1}$$

$d(\boldsymbol{x}, X_k)$ in $D_\beta(\boldsymbol{x}, X_k)$ corresponds to the coffee-house design criterion in [26], and $\beta \cdot d(\boldsymbol{x}, \partial\mathcal{R})$ reflects the phobia of boundary. The design constructed using the above method is denoted as the boundary-phobic coffee-house (BPCH) design.

Algorithm 2: BPCH design

Input: $X^{\mathrm{ori}}, n_{\max}, \mathcal{R}_N$
$X_k \leftarrow X^{\mathrm{ori}}$;
while $|X_k| < n_{\max}$ **do**
  $\boldsymbol{x}^* \leftarrow \arg\max_{\boldsymbol{x} \in \mathcal{R}_N \setminus X_k} D_\beta(\boldsymbol{x}, X_k)$;
  $X_k \leftarrow X_k \cup \boldsymbol{x}^*$;
end
Output: $X_{n_{\max}}^{\mathrm{BPCH}} = X_k$

The pseudo-code of BPCH design are summarized in Algorithm 2. X^{ori} is the original design needed to start the algorithm, $n_{\max}$ is large enough to obtain a desired design, and $\mathcal{R}_N$ is a subset of N point in $\mathcal{R}$, with $N \gg n_{\max}$, which is well spread over $\mathcal{R}$. In this paper,

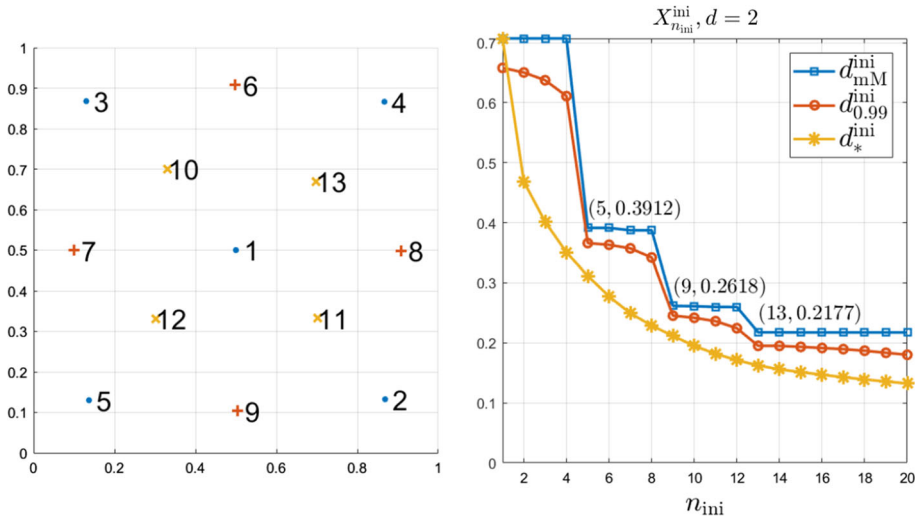


Fig. 7 Examples of initial designs in 2 dimensions

we set $\beta = 2\sqrt{2d}$ (chosen in [37] by trial and error), $d(\boldsymbol{x}, \partial\mathcal{R}^{\text{ini}}) = \min\{\|\boldsymbol{x}\|_\infty, \|\mathbf{1} - \boldsymbol{x}\|_\infty\}$, and $d(\boldsymbol{x}, \partial\mathcal{R}^{\text{seq}}) = 1/2 - \|\boldsymbol{x} - \mathbf{1}/2\|$, where $\|\cdot\|_\infty$ denotes the infinite norm. X^{ori} is set to be $\{\mathbf{1}/2\}$ in the initial and sequential designs. In addition, for the initial design, X^{ori} can be set as some user-defined starting points, which is not permitted in most DIRECT-type algorithms [14]. Because the $\max_{\boldsymbol{x}\in\mathcal{R}} d(\boldsymbol{x}, X_n)$ in Eq. (3.8) is always obtained at $\boldsymbol{x} \in \partial\mathcal{R}$, we set $\mathcal{R}_N = \mathcal{R}_{\text{U}} \cup \mathcal{R}_{\text{C}}$ to better represent $\mathcal{R}$, where $\mathcal{R}_{\text{U}}$ is uniformly sampled in $\mathcal{R}$ and $\mathcal{R}_{\text{C}}$ is the complementary points on $\partial\mathcal{R}$. For uniformity of $\mathcal{R}_N$, $\mathcal{R}_{\text{C}}$ should be much less than $\mathcal{R}_{\text{U}}$. In this paper, we set $\mathcal{R}^{\text{ini}}_{\text{C}}$ as the 2^d vertices of $\mathcal{R}^{\text{ini}}$ and $\mathcal{R}^{\text{seq}}_{\text{C}}$ as 10^4 uniform samples on the hyper-sphere $\partial\mathcal{R}^{\text{seq}}$, set $\mathcal{R}^{\text{ini}}_{\text{U}}$ and $\mathcal{R}^{\text{seq}}_{\text{U}}$ as 10^5 uniform samples in $\mathcal{R}^{\text{ini}}$ and $\mathcal{R}^{\text{seq}}$, respectively.

Figure 7 provides the two-dimensional examples of $X^{\text{ini}}_{n_{\text{ini}}}$ and $d^{\text{ini}}_{\text{mM}}$ in sequence, and Fig. 8 shows the counterparts of $X^{\text{seq}}_{n_{\text{seq}}}$. Compared with Fig. 5, the BPCH designs have larger d_{mM} than the theoretical ones under the condition of same number of points. Nevertheless, the BPCH design has many advantages. First, it generates acceptable designs in relatively short periods, even in high dimensions. Second, setting X^{ori} provides the flexibility to arrange points. Third, the nested structure ($X^{\text{BPCH}}_{n_1} \subseteq X^{\text{BPCH}}_{n_2}, n_1 \le n_2$) permits us to choose a suitable number for the prefix of $X^{\text{BPCH}}_{n_{\max}}$. We next give some experiential guidance for choosing n_{ini} and n_{seq}.

In the right panel of Fig. 7 we can see that when $n_{\text{ini}} = 5, 9, 13$, there are steep drops in d_{mM}. We plotted X^{ini}_5, X^{ini}_9, and X^{ini}_{13} using different markers in the left panel of Fig. 7 and found that those points with specific numbers have relative symmetry. Furthermore, there are steep drops of d_{mM} from $n_{\text{seq}} = 5, \ldots, 8$ in the right panel of Fig. 8, and d_{mM} remained unchanged for many integers after $n_{\text{seq}} = 8$. Similar things occurred at $n_{\text{seq}} = 13$. X^{seq}_5, X^{seq}_8, and X^{seq}_{13} are plotted with different markers in the left panel of Fig. 8, and we believe that X^{seq}_8 and X^{seq}_{13} are more symmetrical than X^{seq}_5. Based on the above analyses, we suggest choosing the appropriate n_{ini} and n_{seq} to meet the following two empirical conditions: (i) d_{mM} has a steep drop at $n_{\text{ini}}(n_{\text{seq}})$, or (ii) d_{mM} is unchanged for several integers after $n_{\text{ini}}(n_{\text{seq}})$. We will analyze the influence of n_{ini} and n_{seq} in Sect. 5.

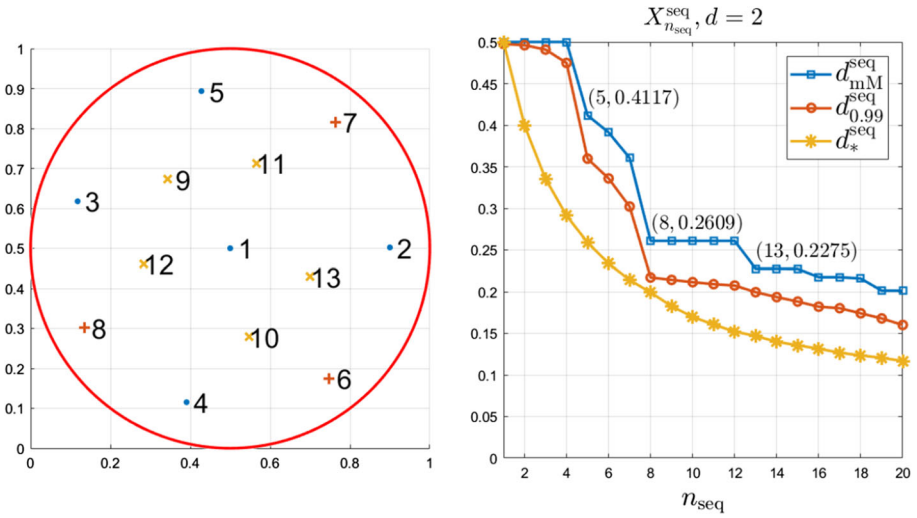


Fig. 8 Examples of sequential designs in 2 dimensions

4.2 Covering and overlapping rates

From the demonstration of SCABALL in Fig. 3(f), we draw the following three facts: (i) sub-regions scattered from the same region, namely $\mathcal{R}_i^{(k+1)}, i \in I_{j^*}^{(k)}$, may overlap; (ii) sub-regions scattered from different regions, namely $\mathcal{R}_i^{(k)}$ and $\mathcal{R}_j^{(k)}$, may overlap; (iii) the union of the sub-regions exceeds the original region, namely $\bigcup_{i \in I_{j^*}^{(k)}} \mathcal{R}_i^{(k+1)} \nsubseteq \mathcal{R}_{j^*}^{(k)}$. Due to these three facts, $\mathcal{R}_i^{(k+1)}$ can be contracted to obtain a more favorable lower boundary defined in Eq. (3.7). We will show that appropriate contraction can accelerate the convergence of SCABALL with little influence on the global convergence.

For a design X_n in $\mathcal{R}$, we define the covering rate function of r as

$$\text{CR}(r) \triangleq \text{CR}(r; X_n, \mathcal{R}) = \frac{\text{vol}(\bigcup_{\boldsymbol{x}_i \in X_n} \mathcal{B}(\boldsymbol{x}_i, r) \cap \mathcal{R})}{\text{vol}(\mathcal{R})}. \tag{4.2}$$

Clearly, $\text{CR}(r)$ is non-decreasing with r. Define the α-distance by

$$d_\alpha(X_n) \triangleq d_\alpha(X_n; \mathcal{R}) = \text{CR}^{-1}(\alpha) = \inf\{r | \text{CR}(r) \geq \alpha\}. \tag{4.3}$$

$d_\alpha(X_n)$ is the minimum radius r that $\bigcup_{\boldsymbol{x}_i \in X_n} \mathcal{B}(\boldsymbol{x}_i, r)$ covers at least $100\alpha\%$ of $\mathcal{R}$, and $d_1(X_n) = d_{\text{mM}}(X_n)$, obviously. We denote $d_\alpha^{\text{ini}} \triangleq d_\alpha(X^{\text{ini}}_{n_{\text{ini}}})$, $d_\alpha^{\text{seq}} \triangleq d_\alpha(X^{\text{seq}}_{n_{\text{seq}}})$, and $d^{\text{ini}}_{0.99}$, $d^{\text{seq}}_{0.99}$ are plotted in the right panel of Figs. 7 and 8, and they are extremely close to $d^{\text{ini}}_{\text{mM}}$ and $d^{\text{seq}}_{\text{mM}}$ in 2 dimensions. However, $d^{\text{ini}}_{0.99}(d^{\text{seq}}_{0.99})$ is much smaller than $d^{\text{ini}}_{\text{mM}}(d^{\text{seq}}_{\text{mM}})$ in higher dimensions (see Figs. 9 and 10). This can be interpreted as that d_{mM} being significantly reduced by ignoring extreme points in high dimensions. This property is also termed "do not try to cover the vertices" in [45]. Additionally, the numerical study of [28] confirmed that X_n^{BPCH} with proper β has strong properties when measured by d_α.

In practice, we would take the sub-region $\mathcal{R}_i = \mathcal{B}(\boldsymbol{x}_i, d_\alpha)$ for some $\alpha < 1$. On the one hand, d_α is smaller than d_{mM}, which provides a better lower bound and accelerates the algorithm. On the other hand, the global convergence in Theorem 3.1 may not hold. It is

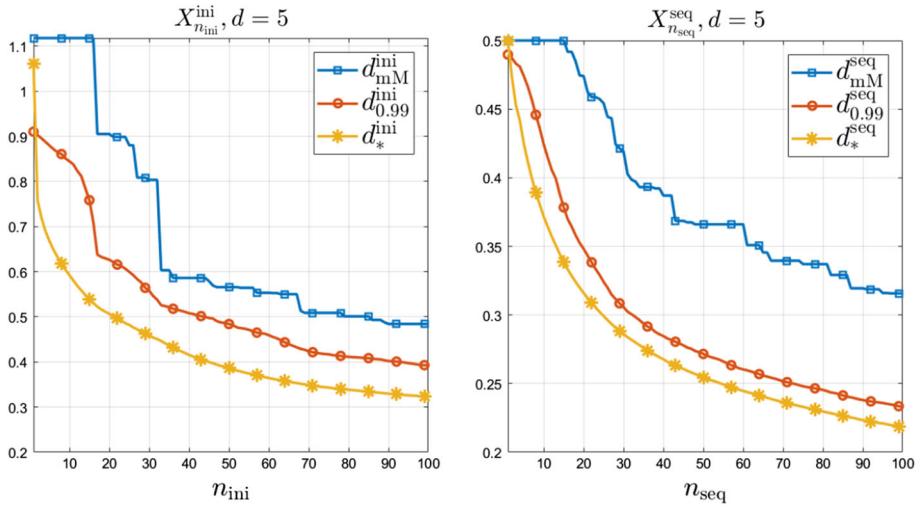


Fig. 9 $d_{\text{mM}}, d_{0.99}, d_*$ of $X^{\text{ini}}_{n_{\text{ini}}}$, and $X^{\text{seq}}_{n_{\text{seq}}}$ in 5 dimensions

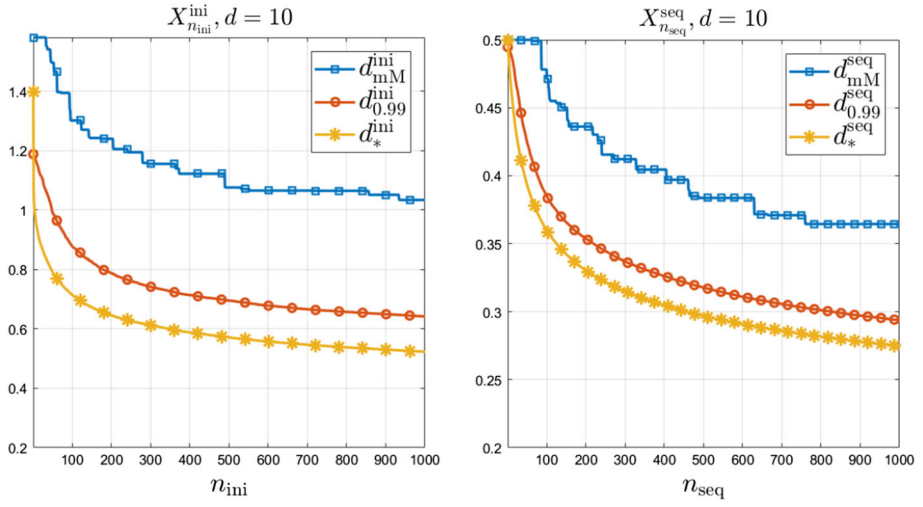


Fig. 10 $d_{\text{mM}}, d_{0.99}, d_*$ of $X^{\text{ini}}_{n_{\text{ini}}}$, and $X^{\text{seq}}_{n_{\text{seq}}}$ in 10 dimensions

importance to choose an appropriate α to balance efficiency and global convergence. To do so, we define the overlapping rate function of r for a design X_n in $\mathcal{R}$ using

$$\text{OR}(r) \triangleq \text{OR}(r; X_n, \mathcal{R}) = \frac{\text{vol}(\bigcup_{\boldsymbol{x}_i, \boldsymbol{x}_j \in X_n, \boldsymbol{x}_j \neq \boldsymbol{x}_i} (\mathcal{B}(\boldsymbol{x}_i, r) \cap \mathcal{B}(\boldsymbol{x}_j, r)) \cap \mathcal{R})}{\text{vol}(\mathcal{R})}. \tag{4.4}$$

$\text{OR}(r)$ is also non-decreasing and $\text{OR}(r) \leq \text{CR}(r)$. To obtain an appropriate contraction, we define the star-distance as

$$d_*(X_n) \triangleq d_*(X_n; \mathcal{R}) = \inf\{r | \text{OR}(r) = 1 - \text{CR}(r)\}. \tag{4.5}$$

We denoted $d_*^{\text{ini}} \triangleq d_*(X_{n_{\text{ini}}}^{\text{ini}})$, $d_*^{\text{seq}} \triangleq d_*(X_{n_{\text{seq}}}^{\text{seq}})$, and plotted them in Fig. 7, 8, 9, and 10. d_* corresponds to the radius such that the non-covering rate is equal to the overlapping rate. If the uncovered region happens to be replenished by the overlapped region exactly, which is impossible, d_* may be the best choice for contraction. In this sense, d_* is an underestimate of the best contracted radius. For flexibility, we introduce a parameter γ, and replace the updating in Eq. (3.13) as

$$\begin{aligned} d(\mathcal{R}_i^{(1)}) = r_i^{(1)} &= d_*^{\text{ini}}, \ \forall i \in I_0^{(0)}, \\ d(\mathcal{R}_i^{(k+1)}) = r_i^{(k+1)} &= 2\gamma d_*^{\text{seq}} \cdot d(\mathcal{R}_{j^*}^{(k)}), \ \forall i \in I_{j^*}^{(k)}. \end{aligned} \tag{4.6}$$

We will analyze the influence of n_{ini}, n_{seq}, and γ in next section.

5 Numerical results

In this section, we first present some numerical analyses of the parameters in SCABALL, then make some comparisons with other DIRECT-type algorithms. We use the GKLS-generator [10] to provide a large number of random test functions. The parameters of the GKLS-generator include (i) the problem dimension d; (ii) the global minimum value f^*; (iii) the number of local minima m; (iv) the radius of the attraction region of the global minimum ρ^*; and (v) the distance from the global minimum to the quadratic function vertex r^*. For more details about the GKLS-generator and related analysis, please refer to [10, 36]. For each dimension, we generated 100 multi-modal and non-differentiable functions for testing. The stopping criteria are as follows: (i) the minimum achieves the specified relative error ($\delta = |(f_{\min} - f^*)/f^*| \leq 0.01$)[1]; and (ii) the number of function evaluations exceeds the limit ($N_{\max} = 10^4 d$). When criterion (ii) is met, we believe the problem is not solved with limited evaluations.

5.1 Choice of parameters

In order to study the influence of n_{ini}, n_{seq}, and γ on SCABALL, we divided them into three levels to perform a full factor experiment. As analyzed in Sect. 4, we set $n_{\text{ini}} = 5, 9, 13$, $n_{\text{seq}} = 5, 8, 13$, and $\gamma = 0.9, 1.1, d_{\text{mM}}/d_*$, respectively, for $d = 2$. The level $\gamma = d_{\text{mM}}/d_*$ corresponds to the uncontracted version of SCABALL, and $d_{\text{mM}}/d_* > 1.1$ for all the designs. The parameter ϵ in Eq. (2.5) is a general parameter in DIRECT-type algorithms to avoid excessive refinement of the local minima. The influence of ϵ is not the focus of our work, thus we fixed $\epsilon = 10^{-4}$ for all DIRECT-type algorithms in this paper, see [14] for more details of ϵ. For every 27 parameter combinations, we uses SCABALL to solve the 100 problems generated by GKLS with parameters

$$f^* = -1, \ m = 5d, \ \rho^* = 1/3, \ r^* = 2/3. \tag{5.1}$$

The number of evaluations (N_{eva}) and the number of solved problems (N_{sol}) are recorded. We performed a profile analysis of each parameter; all the results are classified as profiles according parameter level. As a result, there are 9 profiles each containing 900 results.

[1] Due to the "everywhere dense" convergence, the calculation burden of DIRECT-type algorithms grows geometrically with the dimension, and it is difficult to achieve a finer solution [14]. Therefore, we set the relative error $\delta = 0.1$ for $d \geq 6$ to reduce calculations.

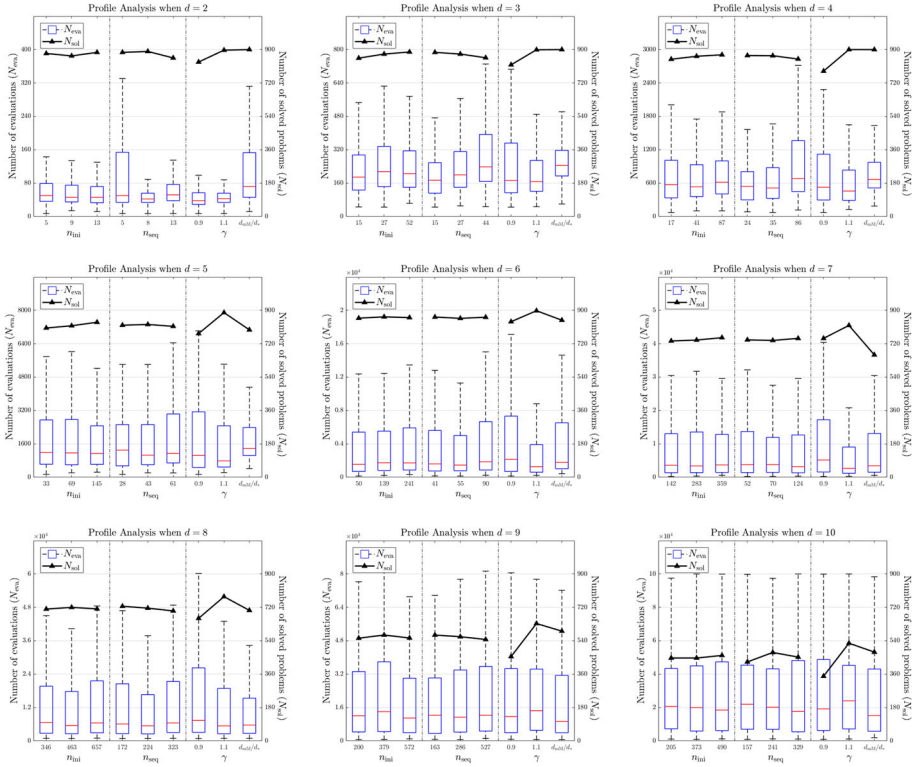


Fig. 11 Profile analyses of n_{ini}, n_{seq}, and γ in SCABALL

For $d = 2$, we made box-plots of N_{eva} and line-plots of N_{sol} in Fig. 11(a). The outliers of N_{eva} are omitted for clarity. It can be seen that $n_{\text{ini}} = 13$, $n_{\text{seq}} = 8$ are good choices for $d = 2$, considering that they both have less N_{eva} and more N_{sol}. Notably, that $N_{\text{eva}} = 2 \times 10^4$ is enough for $d = 2$ and $\delta = 0.01$, this can be verified by the magnitude of N_{eva} in the solved problems. Therefore, the unsolved problems in SCABALL are caused by the loss of global convergence. For the choice of γ, we should balance efficiency and global convergence. On the one hand, N_{eva} increases with γ, which means that a small γ accelerates convergence. On the other hand, N_{sol} decreases significantly when γ is too small. To balance efficiency and global convergence, we chose $\gamma = 1.1$. Note that the N_{sol} of the profile $\gamma = 1.1$ is 897. This illustrates that appropriate contraction can accelerate the convergence of SCABALL with little influence on global convergence.

Regrettably, mM designs do not have nested structures in different dimensions, which means we need to construct $X^{\text{ini}}_{n_{\text{ini}}}(X^{\text{seq}}_{n_{\text{seq}}})$ for each dimension, and so does the analysis above. The profile analyses of other dimensions are shown in Fig. 11. We summarize some general rules as follows. The sensitivity of the SCABALL algorithm to n_{ini}, n_{seq}, and γ increases successively. The effect of n_{ini} and n_{seq} on N_{sol} is not obvious, while they have some effect on N_{eva}. This means that the number of designs only influences the convergence rate. The best n_{ini} and n_{seq} we found grow with the dimension, and n_{ini} increases faster than n_{seq}. Conversely, γ has significant influence on both N_{eva} and N_{sol}. In general, both N_{eva} and N_{sol} increase with γ, but too large an N_{eva} may lead to a decrease in N_{sol}. This phenomenon begins

to occur at $d = 5$ and higher dimensions. In high dimensions, the contraction of SCABALL is necessary to reduce the calculations. In addition, too small a γ makes N_{eva} spread more extreme, which means the algorithm is becoming unstable. Therefore, the choice of γ must be careful. Fortunately, $\gamma = 1 \sim 1.2$ is always a good choice for a fairly wide range of dimensions. This confirms that the d_* that we defined in Eq. (4.5) is a good underestimate of the best contracted radius. Considering that the construction of mM design is stochastic, we provide the empirical formulae to choose $n_{\text{ini}}, n_{\text{seq}}$, and γ, as follows:

$$n_{\text{ini}} = 5d^2 - 7, \quad n_{\text{seq}} = d^2 + 5d - 6, \quad \gamma = 1.1. \tag{5.2}$$

Since the SCABALL algorithm is not sensitive to n_{ini} and n_{seq}, one can choose suitable numbers of design points near the formulae (5.2) by referring to the two suggestions at the end of Sect. 4.1. Although constructing $X^{\text{ini}}_{n_{\text{ini}}}$ and $X^{\text{seq}}_{n_{\text{seq}}}$ is time consuming, it is only implemented once, and consequently will not influence the efficiency of SCABALL.

5.2 Comparisons with original partition methods

In this subsection, we first compare the performance of SCABALL and some DIRECT-type algorithms with other kinds of partitions. The DIRECT, ADC, DISIMPL-V, DISIMPL-C, and BIRECT algorithms are under consideration; we call these as the original partition methods. The details of the original partition methods are introduced in Sect. 2, and the parameters in SCABALL refer to Eq. (5.2). We used these algorithms to solve the problems generated by GKLS with parameters in Eq. (5.1) and made the line-plots of N_{eva}-N_{sol}, which is also called as the operational characteristics, in Fig. 12. The operational characteristics is a common used visual comparison of deterministic algorithms, more details can be found in Chapter 3 of [42] and [36]. All computations were performed on Intel(R) Core(TM) i7-9750 H 2.60GHz processors running Matlab R2017b. The DIRECT-type algorithms mentioned here and later are all implemented using the dynamic version of the MATLAB toolbox in DIRECTGO v1.0.0 [40].

For $d = 2 \sim 5$, the efficiency of the SCABALL algorithm is superior among all other DIRECT-type algorithms with different partitions. Most random problems can be solved by SCABALL with the minimum N_{eva}, but SCABALL is less efficient in solving extreme problems. This is reflected at the top of the N_{eva}-N_{sol} line, which is dragged to the right when N_{sol} is close to 100, and this is obvious when $d = 3, 4$. It means that the SCABALL algorithm needs more evaluations than other algorithms to solve those extreme problems.

To further analyze the efficiency of SCABALL, we use GKLS to generate a new class of test functions with hard parameters:

$$f^* = -1, \ m = 5d, \ \rho^* = 0.2, \ r^* = 0.8. \tag{5.3}$$

The parameters in Eq. (5.3) have smaller ρ^* and larger r^* comparing with Eq. (5.1), which make it hard for the algorithm to locate the global minimum. For convenience, we call the functions generated by GKLS with parameters (5.1) and (5.3) as simple class and hard class respectively. We did the similar experiments with the hard class of functions, and the results are displayed in Fig. 13.

It turns out that the hard class problems need more evaluations to solve, and the SCABALL algorithm is not outstanding in solving them. We believe there are two reasons for this phenomenon. One is that we chose the parameters in SCABALL by the experiments on simple class problems, these parameters may not suitable for solving the hard ones. To achieve an outstanding efficiency, the parameters in SCABALL must be selected for a specific class

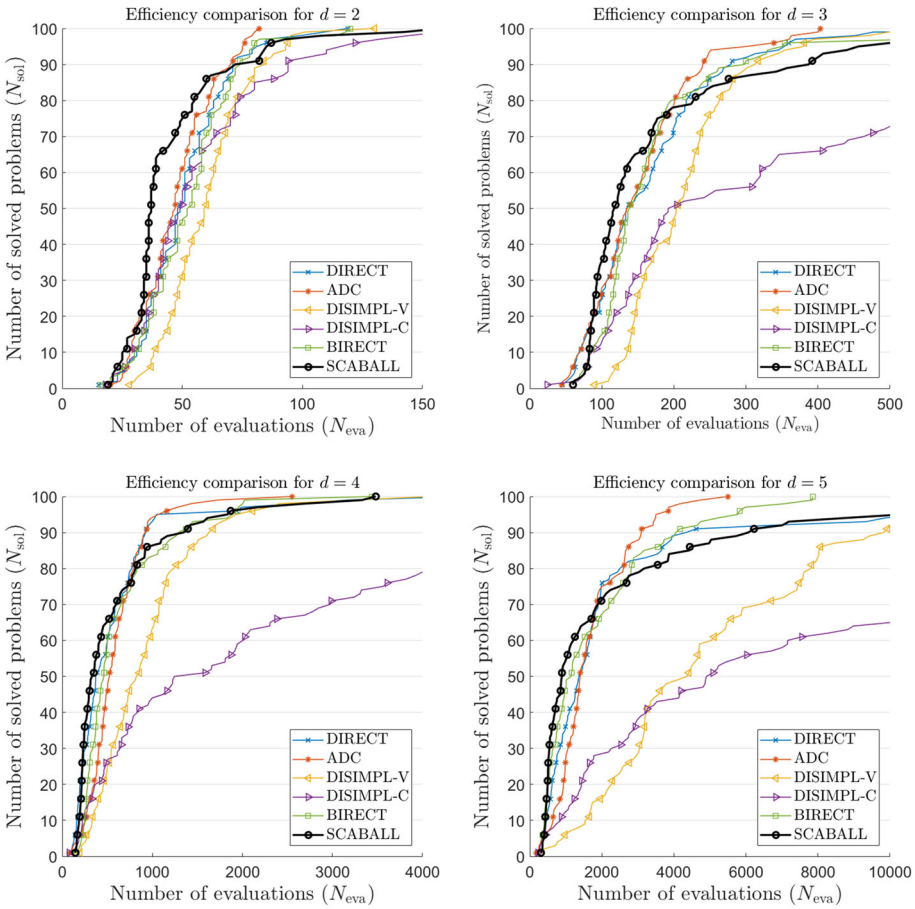


Fig. 12 Optional characteristics of original partition methods on simple class GKLS problems

of problems, otherwise, the efficiency will become mediocre. The other reason is that the DIRECT-type algorithms is greatly influenced by the groups of $d(\mathcal{R}_i)$. According to the updating in Eq. (3.14), the $d(\mathcal{R}_i)$ in SCABALL are neat. This perspective is illustrated by Fig. 14. We ran all the DIRECT-type algorithms until the number of $\mathcal{R}_i$ was 100, and created scatter diagrams of $d(\mathcal{R}_i)$ and $f(\mathcal{R}_i)$. It is obvious that SCABALL can decrease the number of $d(\mathcal{R}_i)$-based groups of $\mathcal{R}_i$, and the reduction would bias the SCABALL towards a faster convergence to local minima [9]. This also explains the good performance of SCABALL on the relatively simple class problems with limited evaluations.

5.3 Comparisons with improved and hybrid methods

Next, we chose some variants of DIRECT-type algorithms to analyze their efficiency in higher dimensions, including DIRECT-l [9], PLOR [25], DIRECT-rev [13], Gb-BIRECT and BIRMIN [32] algorithms. These algorithms have good performance among DIRECT-type algorithms, see the numerical comparisons in the Sect. 4.1 of [40]. The DIRECT-l, PLOR,

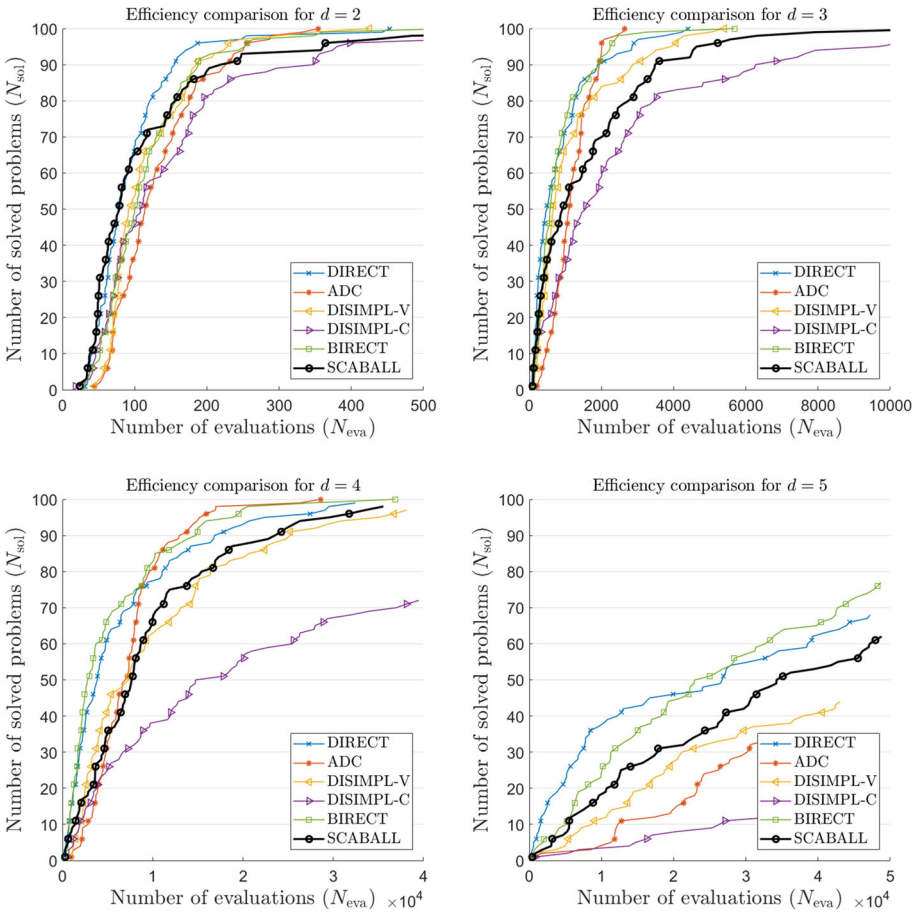


Fig. 13 Optional characteristics of original partition methods on hard class GKLS problems

and Gb-BIRECT algorithms change the selection of POR to make the algorithms more efficient, and we call these improved methods. The DIRECT-l decreases the $d(\mathcal{R}_i)$-based groups by using the infinity norm, similar to SCABALL, it is locally biased. The Gb-BIRECT introduces a phase that constrains the selection of POR to large sub-regions, which is globally biased. The PLOR balances the global and local search by choosing only the two with minimal and maximal $d(\mathcal{R}_i)$. We choose these improved methods to compare the influence of locally and globally biased strategies. The DIRECT-rev and BIRMIN algorithms combine the local optimizers to accelerate convergence, and we call these hybrid methods. They use *fmincon* when some improvement in the best current solution is obtained. The DIRECT-rev also includes a revised partition scheme. We choose these hybrid methods to analyze the influence of the assistance of local optimizer. For better comparison, the corresponding original partition methods, DIRECT and BIRECT, are also under consideration.

The comparison results for high dimensional GKLS functions of simple class are shown in Fig. 15. As for the high dimensional GKLS functions of hard class, they are too difficult to be solved in limited evaluations. In fact, all the algorithms we selected cannot solve even half of them, thus we do not display the results. It can be seen that, none of DIRECT-type algorithms,

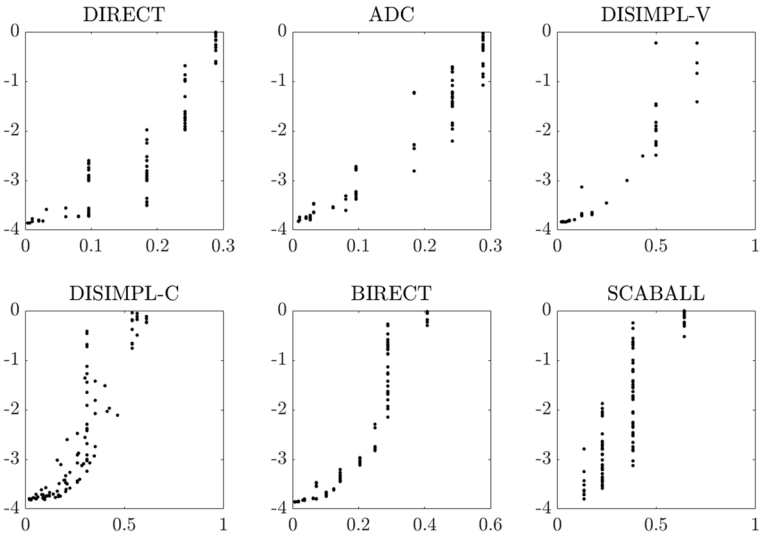


Fig. 14 $d(\mathcal{R})$–$f(\mathcal{R})$ diagrams of DIRECT-type algorithms

even combined with local optimizers, are immune to the "curse of dimensionality," they cannot solve all simple problems with limited evaluations in higher dimensions. Among all these methods, the SCABALL algorithm performs well on this class of problems. In this experiment, it is even comparable with the two hybrid methods. The DIRECT, DIRECT-rev, and BIRMIN are also in good performance, while the DIRECT-l and PLOR are the least efficient. We believe that the local strategies of DIRECT-l and PLOR are not suitable to this class of functions.

To increase the variety of test functions, we present the comparison results of DIRECT-type algorithms on the box-constrained problems in the DIRECTLib v1.3 [38]. DIRECTLib contains a large number of various objective functions, including uni-modal and multi-modal, convex and non-convex problems, and the dimensionality covers 2-10. The key characteristics of these test problems are listed in Table 2. Figure 16 displays the optional characteristics on all 129 problems; the horizontal axes are logarithmic for better illustration.

It turns out that the hybrid methods are the most efficient, especially for those difficult problems needing more evaluations. Obviously, the local optimizer greatly improved the efficiency of DIRECT-type algorithms. The influence of POR selection is also shown in Fig. 16. The DIRECT-l is a local version of DIRECT. From the general trend, the N_{eva}-N_{sol} line of DIRECT-l passes through that of DIRECT from the left. It means that the local strategy helped DIRECT-l to solve more simple problems with less evaluations. On the contrary, Gb-BIRECT is a global version of BIRECT. And the N_{eva}-N_{sol} line of Gb-BIRECT passes through that of BIRECT from the bottom. It means that the global phase helped Gb-BIRECT to solve more total problems than BIRECT. The PLOR is a balanced and simplified version of DIRECT, and it is one of the most efficient algorithm except for the hybrid methods. This results is different from the previous experiments, which indicate that the strategy of PLOR is efficient but lack of stability. Among all original partition methods and improved methods, the efficiency of SCABALL algorithm is mediocre, but it can solve the most DIRECTLib problems in limited evaluations. Due to the variety of DIRECTLib problems, this shows the robustness of SCABALL to some extent.

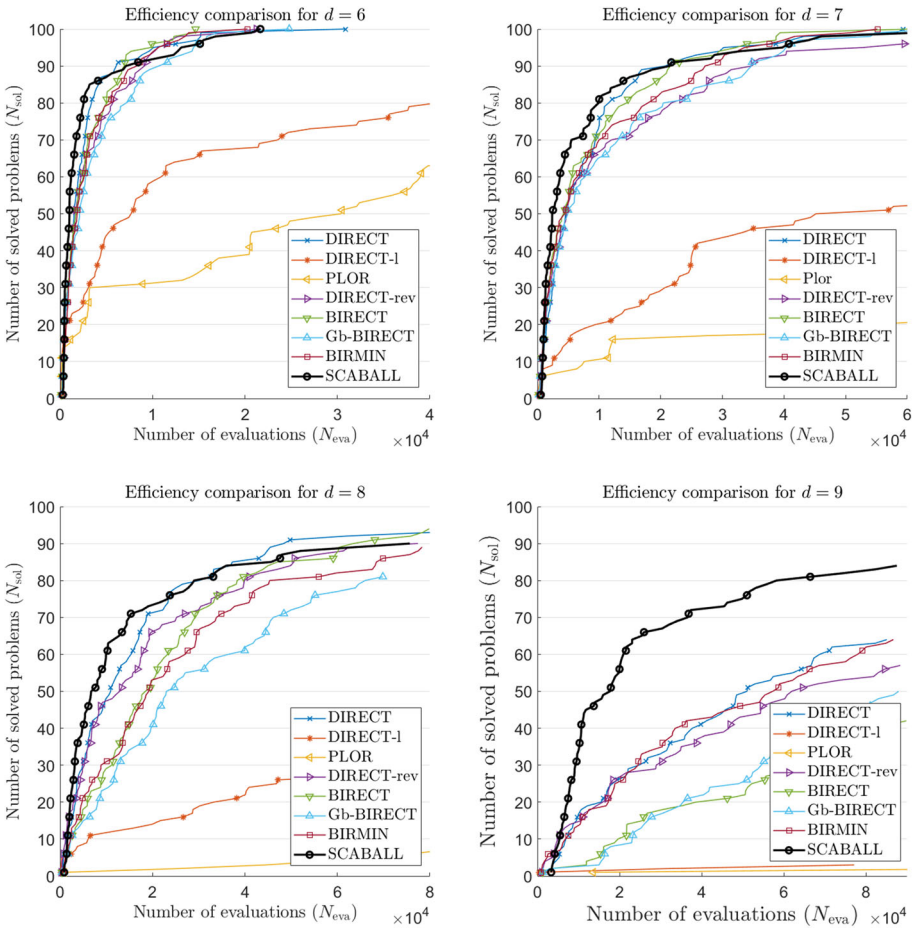


Fig. 15 Optional characteristics of improved and hybrid methods on simple class GKLS problems

6 Conclusion and further discussion

In this paper, we introduced a new SCABALL algorithm for derivation-free optimization problems. SCABALL is a DIRECT-type algorithm with a new partition method. It does not focus on dividing the region of interest into specific geometry, but rather scatters several balls to cover it. Then, to achieve better coverage, approximate mM designs were constructed by boundary-phobic coffee-house design, and the partition radii were contracted by analyzing the covering and overlapping rate. The parameters in SCABALL were analyzed using a numerical method, and the empirical choices were given by formulae. Finally, we did plenty numerical experiments to compare the efficiency of SCABALL and other DIRECT-type algorithms. The numerical results show that the SCABALL algorithm is locally biased, which can solve most simple problems efficiently but is not outstanding in solving hard problems. In addition, the SCABALL algorithm is robust to some extend.

There are also weaknesses in SCABALL. One is that the performance of SCABALL is greatly influenced by the mM designs; therefore, the construction of $X_{n_{\text{ini}}}^{\text{ini}}$ and $X_{n_{\text{seq}}}^{\text{seq}}$ and the

Table 2 Key characteristics of the DIRECTlib test problems

No.	Name	d	Global minimum f^*	Optimization domain
1,2,3	Ackley	2,5,10	0	$[-18, 47]^d$
4,5,6	Alpine	2,5,10	-2.80813118^d	$\times[2^{1/i}, 8 + 2^{1/i}]$
7	Beale	2	0	$[-4.5, 4.5]^d$
8	Bohachecsky1	2	0	$[-55, 145]^d$
9	Bohachecsky2	2	0	$[-55, 145]^d$
10	Bohachecsky3	2	0	$[-55, 145]^d$
11	Booth	2	0	$[-10, 10]^d$
12	Branin	2	0.397887358	$[-5, 10] \times [0, 15]$
13	Bukin6	2	0	$[-15, 5] \times [-3, 3]$
14	Colville	4	0	$[-10, 10]^d$
15	Cross in Tray	2	-2.062611871	$[0, 10]^d$
16	Crosslegtable	2	-1	$[-10, 15]^d$
17,18,19	Csendes	2,5,10	0	$[-10, 25]^d$
20	Damavandi	2	0	$[0, 14]^d$
21,22,23	Deb01	2,5,10	-1	$[-0.55, 1.45]^d$
24,25,26	Deb02	2,5,10	-1	$[0.225, 1.225]^d$
27,28,29	Dixon and Price	2,5,10	0	$[-10, 10]^d$
30	Drop wave	2	-1	$[-4, 6]^d$
31	Easom	2	-1	$[-50, 100] \times [-33.33, 200]$
32	Eggholder	2	-959.6406627	$[-512, 512]^d$
33	Goldstein and Price	2	3	$[-1.1, 2.9]^d$
34,35,36	Griewank	2,5,10	0	$\times[-\sqrt{600i}, 600/\sqrt{i}]$
37	Hartman3	3	-3.862782148	$[0, 1]^d$

Table 2 continued

No.	Name	d	Global minimum f^*	Optimization domain
38	Hartman6	6	-3.322368011	$[0, 1]^d$
39	Holder	2	-19.20850257	$[-10, 10]^d$
40	Hump	2	-1.031628453	$[-5, 5]^d$
41	Langermann	2	-4.155809292	$[0, 10]^d$
42,43,44	Levy	2,5,10	0	$[-10, 10]^d$
45	Matyas	2	0	$[-5.5, 14.5]^d$
46	McCormick	2	-1.913222955	$[-1.5, 4] \times [-3, 4]$
47,48,49	Michalewicz	2,5,10	-1.801303410, -4.687658179, -9.660151716	$[0, \pi]^d$
50	Perm	4	0	$\times[-i, i]$
51,52,53	Pinter	2,5,10	0	$[-5.5, 14.5]^d$
54	Powell	4	0	$[-4, 5]^d$
55	Power Sum	4	0	$\times[1, 4 + 2^{1/i}]$
56,57,58	Problem01	2,5,10	0	$[-2, 3]^d$
59,60,61	Problem02	2,5,10	0	$[0, 5]^d$
62,63,64	Problem03	2,5,10	0	$[-1, 2]^d$
65,66,67	Problem04	2,5,10	0	$[0, 5]^d$
68,69,70	Problem05	2,5,10	0	$[-4, 3]^d$
71,72,73	Problem06	2,5,10	0	$[-3, 0]^d$
74,75,76	Problem07	2,5,10	0	$[0, 7]^d$
77,78,79	Problem08	2,5,10	0	$[-30, 20]^d$
80,81,82	Problem09	2,5,10	0	$[-3, 7]^d$
83,84,85	Qing	2,5,10	0	$[-500, 500]^d$
86,87,88	Rastrigin	2,5,10	0	$\times[-5 \times 2^{1/i}, 7 + 2^{1/i}]$

Table 2 continued

No.	Name	d	Global minimum f^*	Optimization domain
89,90,91	Rosenbrock	2,5,10	0	$\times[-5/2^{1/i}, 10 \times 2^{1/i}]$
92,93,94	Rotated Hyper Ellipsoid	2,5,10	0	$[-35, 96]^d$
95,96,97	Schwefel	2,5,10	0	$\times[-500 + 100/\sqrt{i}, 500 - 40/\sqrt{i}]$
98	Shekel10	4	-10.53640982	$[0, 10]^d$
99	Shekel5	4	-10.15319968	$[0, 10]^d$
100	Shekel7	4	-10.40294057	$[0, 10]^d$
101	Shubert	2	-186.7309088	$[-10, 10]^d$
102,103,104	SineEnvelope	2,5,10	$-2.6535768(d-1)$	$[-100, 100]^d$
105,106,107	Sphere	2,5,10	0	$[-2.75, 7.25]^d$
108,109,110	Styblinski Tang	2,5,10	$-39.1661657d$	$\times[-5, 5 + 3^{1/i}]$
111,112,113	Sum Square	2,5,10	0	$[-5.5, 14.5]^d$
114,115,116	Sum of Different Powers	2,5,10	0	$[-0.55, 1.45]^d$
117	Trefethen	2	-3.306868647	$[-2, 2]^d$
118,119,120	Trid	2,5,10	$-d(d+4)(d-1)/6$	$[-100, 100]^d$
121,122,123	Vincent	2,5,10	$-d$	$[0.25, 10]^d$
124,125,126	Xinsheyangn3fcn	2,5,10	-1	$[-11, 29]^d$
127,128,129	Zakharov	2,5,10	0	$[-1.625, 13.38]^d$

*$\times[a_i, b_i] \triangleq [a_1, b_1] \times [a_2, b_2] \times \cdots \times [a_d, b_d]$

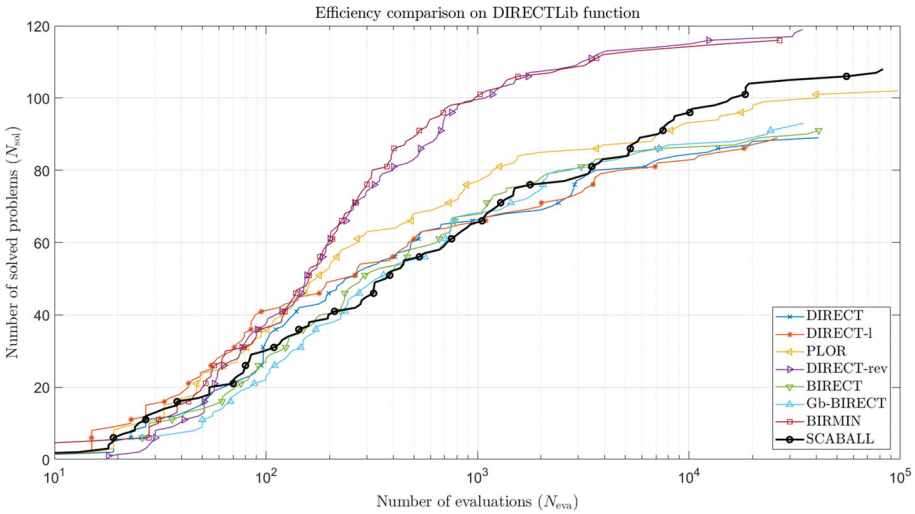


Fig. 16 Optional characteristics of improved and hybrid methods on DIRECTlib problems

choice of n_{ini}, n_{seq}, and γ would be important. The other is that the mM designs do not have nested structures in different dimensions, one should construct two kinds of mM design for each dimension, which makes it difficult to start the SCABALL algorithm.

Further more, there are several directions left for further study. One is the improved scheme and the hybrid method of SCABALL. Since SCABALL is locally biased, the global phase could be introduced like Gb-BIRECT, and the local optimizer can also combined with SCABALL. Besides, the adaptive parameter tuning of SCABALL is also worth studying. The efficiency of SCABALL could be improved if γ and ϵ can be tuned during iterations. Third, the flexibility of SCABALL needs to be exploited. As mentioned in Sect. 4.1, SCABALL can start on some user-defined points. This flexibility makes it possible to optimize the objective function with the help of priors.

Acknowledgements This work was supported by the National Natural Science Foundation of China (No. 11771450, 61803376) and the National Numerical Wind Tunnel Project (NNW2019ZT7-B23). We would like to thank the anonymous reviewers for their constructive comments, which made the article enriched and rigorous.

Declarations

Conflict of interest The authors have no competing interests to declare that are relevant to the content of this article. The datasets generated during the current study are available from the corresponding author on reasonable request.

Open Access This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit http://creativecommons.org/licenses/by/4.0/.

References

1. Belfkira, R., Zhang, L., Barakat, G.: Optimal sizing study of hybrid wind/PV/diesel power generation unit. Solar Energy **85**, 100–110 (2011)
2. Beliakov, G.: Interpolation of Lipschitz functions. J. Comput. Appl. Math. **196**(1), 20–44 (2006)
3. Beyhaghi, P., Cavaglieri, D., Bewley, T.R.: Delaunay-based derivative-free optimization via global surrogates, part I: linear constraints. J. Glob. Optim. **66**, 331–382 (2016)
4. Cappellari, M., Verolme, E., van der Marel, R.P., Kleijn, G.A.V., Illingworth, G.D., Franx, M., Carollo, C.M., de Zeeuw, P.T.: The counterrotating core and the black hole mass of IC 1459. Astrophys. J. **578**, 787–805 (2002)
5. Conn, A.R., Scheinberg, K., Vicente, L.N.: Introduction to Derivative-Free Optimization, 1st edn. MPS-SIAM, USA (2009)
6. Cortés, J., Bullo, F.: Coordination and geometric optimization via distributed dynamical systems. SIAM J. Control Optim. **44**(5), 1543–1574 (2005)
7. Cortés, J., Bullo, F.: Nonsmooth coordination and geometric optimization via distributed dynamical systems. SIAM Rev. **51**(1), 163–189 (2009)
8. Ecker, J.G., Kupferschmid, M., Lawrence, C.E., Reilly, A.A., Scott, A.C.H.: An application of nonlinear optimization in molecular biology. Eur. J. Oper. Res. **138**, 452–458 (2002)
9. Gablonsky, J.M., Kelley, C.T.: A locally-biased form of the DIRECT algorithm. J. Glob. Optim. **21**(1), 27–37 (2001)
10. Gaviano, M., Kvasov, D.E., Lera, D., Sergeyev, Y.D.: Algorithm 829: software for generation of classes of test functions with known local and global minima for global optimization. ACM Trans. Math. Softw. **29**, 469–480 (2003)
11. Goldberg, D.E.: Genetic Algorithms in Search, Optimization and Machine Learning, 1st edn. Addison-Wesley Longman Publishing Co., Inc, USA (1989)
12. Johnson, M.E., Moore, L.M., Ylvisaker, D.: Minimax and maximin distance designs. J. Stat. Plan. Inference **26**(2), 131–148 (1990)
13. Jones, D.R.: DIRECT global optimization algorithm. In: Floudas, C.A., Pardalos, P.M. (eds.) Encycl. Optim., pp. 431–440. Kluwer Academic, Dordrecht (2001)
14. Jones, D.R., Martins, J.R.: The DIRECT algorithm: 25 years later. J. Glob. Optim. **79**, 521–566 (2021)
15. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the Lipschitz constant. J. Optim. Theory Appl. **79**(1), 157–181 (1993)
16. Kennedy, J., Eberhart, R.: Particle swarm optimization. In: Proceedings of ICNN'95-International Conference on Neural Networks, IEEE, vol. 4, pp. 1942–1948 (1995)
17. Kirkpatrick, S., Gelatt, C.D., Vecchi, M.P.: Optimization by simulated annealing. Science **220**(4598), 671–680 (1983)
18. Kramer, O., Ciaurri, D.E., Koziel, S.: Derivative-free optimization. In: Koziel, S., Yang, X.S. (eds.) Computational Optimization, pp. 61–83. Methods and Algorithms, Springer, Berlin Heidelberg, Berlin, Heidelberg (2011)
19. Lin, M.H.: An optimal workload-based data allocation approach for multidisk databases. Data Knowl. Eng. **68**, 499–508 (2009)
20. Liu, H.T., Xu, S.L., Wang, X.F., Wu, J.N., Song, Y.: A global optimization algorithm for simulation-based problems via the extended DIRECT scheme. Eng. Optim. **47**(11), 1441–1458 (2015)
21. Liu, Q., Zeng, J., Yang, G.: MrDIRECT: a multilevel robust direct algorithm for global optimization problems. J. Glob. Optim. **62**, 205–227 (2015)
22. Liu, Q., Yang, G., Zhang, Z., Zeng, J.: Improving the convergence rate of the direct global optimization algorithm. J. Glob. Optim. **67**, 851–872 (2017)
23. Liuzzi, G., Lucidi, S., Piccialli, V.: A DIRECT-based approach exploiting local minimizations for the solution of large-scale global optimization problems. Comput. Optim. Appl. **45**, 353–375 (2010)
24. Mockus, J.: On the pareto optimality in the context of Lipschitzian optimization. Informatica **22**, 521–536 (2011)
25. Mockus, J., Paulavičius, R., Rusakevicius, D., Šešok, D., Zilinskas, J.: Application of reduced-set Pareto-Lipschitzian optimization to truss optimization. J. Glob. Optim. **67**, 425–450 (2017)
26. Müller, W.G.: Coffee-house designs. In: Atkinson, A., Bogacka, B., Zhigljavsky, A. (eds.) Optimum Design 2000, pp. 241–248. Springer, US, Boston, MA (2001)
27. Niederreiter, H.: Random Number Generation and Quasi-Monte Carlo Methods. Society for Industrial and Applied Mathematics, USA (1992)
28. Nogales-Gómez, A., Pronzato, L., Rendas, M.J.: Incremental space-filling design based on coverings and spacings: improving upon low discrepancy sequences. J. Stat. Theory Pract. **15**, 77 (2021)

29. Paulavičius, R., Žilinskas, J.: Simplicial Lipschitz optimization without the Lipschitz constant. J. Glob. Optim. **59**(1), 23–40 (2014)
30. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. J. Glob. Optim. **59**(2), 545–567 (2014)
31. 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. **71**(1), 5–20 (2018)
32. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased BIRECT algorithm with local accelerators for expensive global optimization. Expert Syst. Appl. **144**, 113052 (2020)
33. Pronzato, L.: Minimax and maximin space-filling designs: some properties and methods for construction. J. de la Société Française de Statistique **158**(1), 7–36 (2017)
34. Sergeyev, Y.D., Kvasov, D.E.: Global search based on efficient diagonal partitions and a set of Lipschitz constants. SIAM J. Optim. **16**(3), 910–937 (2006)
35. Sergeyev, Y.D., Strongin, R.G., Lera, D.: Introduction to Global Optimization Exploiting Space-Filling Curves. Springer, New York (2013)
36. Sergeyev, Y.D., Kvasov, D.E., Mukhametzhanov, M.S.: On the efficiency of nature-inspired metaheuristics in expensive global optimization with limited budget. Sci. Rep. **8**(1), 453 (2018)
37. Shang, B., Apley, D.W.: Fully-sequential space-filling design algorithms for computer experiments. J. Qual. Technol. **53**, 173–196 (2020)
38. Stripinis, L., Paulavicius, R.: DIRECTLib—a library of global optimization problems for DIRECT-type methods. (2022). https://doi.org/10.5281/zenodo.5830927
39. Stripinis, L., Paulavičius, R.: An empirical study of various candidate selection and partitioning techniques in the DIRECT framework. J. Glob. Optim. (2022). https://doi.org/10.1007/s10898-022-01185-5
40. Stripinis, L., Paulavičius, R.: DIRECTGO: a new DIRECT-type matlab toolbox for derivative-free global optimization. ACM Trans. Math. Softw. (2022). https://doi.org/10.1145/3559755
41. Stripinis, L., Paulavičius, R., Žilinskas, J.: Improved scheme for selection of potentially optimal hyper-rectangles in DIRECT. Optim. Lett. **12**(7), 1699–1712 (2018)
42. Strongin, R.G., Sergeyev, Y.D.: Global Optimization with Non-convex Constraints: Sequential and Parallel Algorithms, Nonconvex Optimization and Its Applications, vol. 45. Kluwer Academic Publishers, Dordrecht (2000)
43. Tsai, J.F., Li, H.L., Hu, N.Z.: Global optimization for signomial discrete programming problems in engineering design. Eng. Optim. **34**, 613–622 (2002)
44. Xiao, Y., Maier, A.K., Wein, W., Shams, R., Kadoury, S., Drobny, D., Modat, M., Reinertsen, I., Rivaz, H., Chabanas, M., Fortin, M., Machado, I., Ou, Y., Heinrich, M.P., Schnabel, J.A., Zhong, X.: Evaluation of MRI to ultrasound registration methods for brain shift correction: the curious2018 challenge. IEEE Trans. Med. Imaging **39**, 777–786 (2020)
45. Zhigljavsky, A.A., Noonan, J.: Covering of high-dimensional cubes and quantization. SN Oper. Res. Forum (2020). https://doi.org/10.1007/s43069-020-0015-8

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.