J Glob Optim (2016) 66:811–845
DOI 10.1007/s10898-016-0441-5

MSO: a framework for bound-constrained black-box global optimization algorithms

Abdullah Al-Dujaili[1] · **S. Suresh**[1] · **N. Sundararajan**[2]

Received: 28 July 2015 / Accepted: 12 May 2016 / Published online: 21 May 2016

Abstract This paper addresses a class of algorithms for solving bound-constrained black-box global optimization problems. These algorithms partition the objective function domain over multiple scales in search for the global optimum. For such algorithms, we provide a generic procedure and refer to as multi-scale optimization (**MSO**). Furthermore, we propose a theoretical methodology to study the convergence of **MSO** algorithms based on three basic assumptions: (a) local Hölder continuity of the objective function f, (b) partitions boundedness, and (c) partitions sphericity. Moreover, the worst-case finite-time performance and convergence rate of several leading **MSO** algorithms, namely, Lipschitzian optimization methods, multi-level coordinate search, dividing rectangles, and optimistic optimization methods have been presented.

Keywords Global optimization · Black-box functions · Multi-scale · Space-partitioning · Sampling · Lipschitzian · Convergence analysis

1 Introduction

Optimization algorithms aim to find the best solution from a set of solutions for a problem with a(n) (un)constrained objective function. Many decision-making problems for real-world systems can be formulated as optimization problems of objective functions of a set of decision variables modeling and describing such systems. These functions are often multi-variate, non-differentiable, multi-modal, and tedious to be evaluated (see, e.g., [6,17,50]), which makes

This work was supported by Ministry of Education (MoE), Singapore through tier I (No. M4011269) funding.

✉ S. Suresh
ssundaram@ntu.edu.sg

[1] School of Computer Engineering, Nanyang Technological University, Singapore 639798, Singapore

[2] School of Electrical and Electronic Engineering, Nanyang Technological University, Singapore 639798, Singapore

optimizing them exceptionally difficult. Optimization problems have been a recurring topic for centuries and many numerical techniques to solve them have been proposed and widely discussed in the literature (see, e.g., [26,41,42]).

In some optimization problems, decision variables take values constrained in a given range. Such problems, often referred to as *bound-constrained* optimization problems, play a key role in the design of algorithms for general optimization problems because many of these algorithms reduce their solutions to the solution of a sequence of bound-constrained problems. Moreover, bound-constrained optimization problems are present in several practical applications as the decision variables of many real-world systems are often bounded by physical limits [3,48].

In all these algorithms, certain assumptions are made about the objective function being optimized (e.g., its continuity or differentiability). However, these assumptions are not necessarily satisfied by real-world objective functions and often they are impossible to verify. Sometimes, the function is only available through a black box, where one provides a point, representing a candidate solution (a realization of the decision variables), to the black box, which, in return, evaluates the function at that point and gives the corresponding function value. In other words, the only source of information about the function becomes the set of evaluations performed through the black box. Optimizing such functions is often referred to as a *black-box* optimization problem. Usually, a function evaluation requires computational resources, and hence black-box optimization seeks a trade-off between the quality of a solution found versus the evaluation budget, namely the number of function evaluations (i.e., the amount of computational resources used). Besides, most of real-world objective functions are multi-modal for which one might be interested in finding a *global* or *local* optimum. Global optimization algorithms are guaranteed to find the global optimum given enough computational resources, whereas local optimization algorithms are only guaranteed to find a local optimum, regardless of how many computational resources are used. Therefore, it is common to quantify the quality of the found solution as a function of the function evaluation budget.

Consider the *bound-constrained black-box global optimization problem* (**BCBBGOP**) stated as:[1]

$$\begin{array}{ll} \text{maximize} & f(x) \\ \text{subject to} & x \in \mathcal{X},\ \mathcal{X} = \{x \in \mathbb{R}^N : l \leq x \leq u\} \end{array} \tag{1}$$

using a computational budget of n function evaluations, where $f : \mathcal{X} \subset \mathbb{R}^N \to \mathbb{R}$ is a *black-box function on which no information is available* except the objective function values. Let x^* be the (or one) solution to (1) such that $f^* = f(x^*)$; an algorithm for finding any solution x^*, would be one of two types: *passive*, or *sequential*. A passive algorithm $\mathcal{A}_p$ would use its n-evaluation budget at once to evaluate a uniform grid of n points in the objective function's domain $\mathcal{X}$ (also called the *search space*). $\mathcal{A}_p$ then returns $x(n)$, the point in the grid with the best function value among the grid points, as a guess on x^* (see, e.g., [13]). Alternatively, a sequential algorithm $\mathcal{A}_s$ would start by arbitrarily guessing/selecting a point in the search space and evaluating f at it. $\mathcal{A}_s$ then iteratively refining its next guess based on the previously selected points and their corresponding function f values. After n evaluations, $\mathcal{A}_s$ returns $x(n)$: its best guess on x^* (see, e.g., [43]). Since there is no prior knowledge on f, there is no guarantee that $x(n)$ of either $\mathcal{A}_p$ nor $\mathcal{A}_s$ is the solution to the **BCBBGOP** of (1). While passive algorithms are simple and easy, sequential algorithms can achieve solutions of better

[1] As the problem has been formulated as one of maximization, the words optimize, maximize and their conjugated forms are going to be used interchangeably.

quality given the same computational budget. It was shown independently by Ivanov [28] and Sukharev [63] that the evaluation budget needed to find a solution with a desired degree of quality is the same for the best passive and sequential algorithms in the worst case for some functions, whereas for most other functions, the required evlaution budget for a passive algorithm is much greater than that of a sequential algorithm.

A sequential algorithm is performing an *n-round sequential decision-making process under uncertainty* where a future *action* (selecting a candidate solution) depends on past actions (previously selected points) and their *rewards* (observed function values). A principal question is: how can $\mathcal{A}$ identify as quickly as possible the most rewarding action to select, next. Intuitively, $\mathcal{A}_s$ would *explore* the set of possible actions it can take to know more about f and as its knowledge improves, $\mathcal{A}_s$ should increasingly *exploit* its current knowledge by selecting what it believes to be the best action. Clearly, the trade-off between *exploration* and *exploitation* affects $\mathcal{A}_s$'s returned solution. The quality of $\mathcal{A}_s$'s returned solution $x(n)$, as a function of the number of function evaluations n, is evaluated by the following *regret* (or *loss*) measure:

$$r(n) = f^* - f(x(n)) \tag{2}$$

Such measure also serves as an indicator of $\mathcal{A}_s$'s convergence rate with respect to n, as it gauges how efficient $\mathcal{A}_s$ is, in capturing enough information about f to produce a solution with a desired degree of accuracy.

The literature on sequential decision-making problems under uncertainty is rich and large with applicability in various scientific fields (e.g., machine learning, planning, and control). Initial investigations on such problems date back to Thompson in 1933 [64] and Robbins in 1952 [49], where they were known as *multi-armed bandit* problems. Since then, much progress have been made addressing the exploration vs. exploitation dilemma. Many strategies were proposed, studied, and analyzed; such as ϵ-greedy [2], softmax [5], bayesian [57], upper-confidence-bound [2], and optimistic [33] strategies. With respect to **BCBBGOPs**, Piyavskii [43], Shubert [56], and Strongin [59,60] are considered to be the pioneers in applying the framework of sequential decision making to mathematical optimization. Their work has become the basis on which a large existing body of research relies (see e.g., [25,36,42]). A general discussion on sequential algorithms for global optimization has been presented by Archetti and Betrò [1].

In this paper, we address a class of sequential decision-making algorithms that are designed to solve a **BCBBGOP** via carrying out a *hierarchical partitioning* of f's domain $\mathcal{X}$ in search for x^*. An algorithm of such a class adopts a divide-and-conquer tree search for x^* by partitioning the search space $\mathcal{X}$ in a hierarchical fashion as a part of its n-round sequential decision making. Iteratively, it carries out a dynamic expansion of a tree whose nodes represent subspaces of $\mathcal{X}$; with the root corresponding to the entire search space $\mathcal{X}$. This creates a set of partitions of $\mathcal{X}$ over multiple scales (nodes of a specific depth corresponds to a partition of a specific scale). *We refer to this as Multi-Scale Optimization* (***MSO***). The motivation behind introducing the **MSO** framework is of two folds: first, tree-search algorithms have increasingly been gaining popularity (e.g., game of GO [7]); and second, while there is a great deal of literature on tree-search algorithms in various fields (e.g., artificial agents control and planning [4]), where they have been extensively analyzed; minimal research has been carried out in the context of optimization. Several state-of-the-art hierarchical partitioning algorithms for **BCBBGOPs** have been proposed and analyzed independently (see, e.g., [27,29,38,56]). Nonetheless, the theoretical analysis has been primarily concerned with the asymptotic (limit) behavior of the algorithms and their convergence to optimal points

(see, e.g., [16,52,65]). Here, we study the *finite-time* analysis and convergence rate under the **MSO** framework.

We present a theoretical methodology to analyze **MSO** algorithms, based on three assumptions on the objective function structure, and the hierarchical partitioning. Such methodology quantifies the amount of exploration conducted by an **MSO** algorithm over a partition of a specific scale. Consequently, a bound on the regret (2) can be established, which reflects the finite-time behavior of the algorithm. Using these theoretical findings, we establish finite-time analysis for Lipschitzian optimization methods [56], DIviding RECTangles (**DIRECT**) [29], and Multi-level Coordinate Search (**MCS**) [27]. Furthermore, we integrate the analysis of Deterministic Optimistic Optimization (**DOO**) and Simultaneous Optimistic Optimization (**SOO**) in [38] under the **MSO** framework.

This paper is organized as follows. Basic notations and terminology are introduced in Sect. 2. In Sect. 3, a formal introduction of **MSO** framework to solve **BCBBGOPs** is provided. Towards the end of the section, some well-known **MSO** algorithms are discussed. In Sect. 4, we propose a principled approach for analyzing **MSO** algorithms theoretically and demonstrate it on the algorithms discussed in Sect. 3. Section 5 outlines the theoretical contribution and complements it with empirical validation. Towards the end, Sect. 6 summarizes the conclusions from this study.

2 Some preliminaries

In this section, using some of the concepts from [9], we formally define the hierarchy of partitions and the data structure (tree), on the search space $\mathcal{X}$ employed by an **MSO** algorithm.

We denote by $2^{\mathcal{X}}$ the set of all subsets (subspaces, or cells) of $\mathcal{X}$. The size/volume of a subset in $2^{\mathcal{X}}$ is approximated by the function $\sigma : 2^{\mathcal{X}} \to \mathbb{R}^+$. Two elements $\mathcal{X}_i, \mathcal{X}_j$ of $2^{\mathcal{X}}$ are said to be *disjoint* if and only if

$$\mathcal{X}_i \cap \mathcal{X}_j = \beta(\mathcal{X}_i) \cap \beta(\mathcal{X}_j) \tag{3}$$

Here, $\beta(\mathcal{X}_i)$ denotes the boundary of $\mathcal{X}_i$. A subset of $2^{\mathcal{X}}$ is called a *partial partition* of $\mathcal{X}$ if its elements are disjoint and nonempty. A union of a partial partition is called its *support*. A *partition* of $\mathcal{X}$ is a partial partition whose support is $\mathcal{X}$. A set $\mathcal{G} \subseteq 2^{\mathcal{X}}$ is a *hierarchy on* $\mathcal{X}$ if any two elements of $\mathcal{G}$ are either disjoint or nested, i.e.:

$$\mathcal{X}_i \cap \mathcal{X}_j \in \{\beta(\mathcal{X}_i) \cap \beta(\mathcal{X}_j), \mathcal{X}_i, \mathcal{X}_j\} \text{ for any } \mathcal{X}_i, \mathcal{X}_j \in \mathcal{G} \tag{4}$$

Let X and Y be two distinctive elements of a hierarchy $\mathcal{G}$. We say that Y is a child of X and X is the parent of Y if $Y \subseteq X$ for any $Z \in \mathcal{G}$, such that $Y \subseteq Z \subseteq X$, we have $Z = X$ or $Z = Y$. In other words, X is the smallest superset of Y among $\mathcal{G}$ elements. A partition factor $K \in \mathbb{Z}^+$ of a hierarchy $\mathcal{G}$ is the maximum number of children of a parent in $\mathcal{G}$. An element $L \in \mathcal{G}$ is called a leaf of $\mathcal{G}$ if it has no child.

Now, a hierarchy of partitions on $\mathcal{X}$ is formally defined as:

Definition 1 A hierarchy $\mathcal{G}$ on $\mathcal{X}$ is a ***hierarchy of partitions on*** $\mathcal{X}$ if the union of its leaves is a partition of $\mathcal{X}$ and $\mathcal{X} \in \mathcal{G}$.

The term *multi-scale* in **MSO** is derived from the fact that a hierarchy of partitions has a set of partitions at *multiple scales* $h \in \mathbb{Z}_0^+$. Let $\mathcal{G}$ be a hierarchy of partitions on $\mathcal{X}$ created by an **MSO** algorithm. A hierarchy of partitions $\mathcal{G}$ may be represented by a tree structure $\mathcal{T}(\mathcal{G})$ whose nodes correspond to the elements of $\mathcal{G}$ with the root representing the whole space $\mathcal{X}$,

while its edges link every node corresponding to a child in $\mathcal{G}$ to its corresponding parent's node. $\mathcal{T}$ is referred to as a *tree on* $\mathcal{X}$ if it represents a hierarchy of partitions on $\mathcal{X}$. It is possible to index a node in $\mathcal{T}$ (and subsequently a cell of $\mathcal{X}$) by one or more integer attribute(s). For example, with a partition factor of K, a node can be indexed by its depth/scale h and an index i where $0 \leq i \leq K^h$ as (h, i) which corresponds to a cell/subspace $\mathcal{X}_{h,i} \subset \mathcal{X}$ and possesses up to K children nodes $\{(h+1, i_k)\}_{1 \leq k \leq K}$ such that:

$$\mathcal{X} = \cup_{i \in \{0,\ldots,K^h-1\}} \mathcal{X}_{h,i} \tag{5}$$

$$\mathcal{X}_{h,i} \cap \mathcal{X}_{h,j} = \beta(\mathcal{X}_{h,i}) \cap \beta(\mathcal{X}_{h,j}), \quad i \neq j \tag{6}$$

$$\mathcal{X}_{h,i} = \cup_{1 \leq k \leq K} \mathcal{X}_{h+1,i_k}, \quad \forall h \in \mathbb{Z}_0^+ \tag{7}$$

Nodes can be indexed and grouped by any of their attributes such as depth, and size. For example, $S = \{i \in \mathcal{T} : \sigma(i) = s\}$ is the set of all nodes in the tree $\mathcal{T}$ of size s. Note that for any two elements $i, j \in S$, $i \cap j = \beta(i) \cap \beta(j)$. The set of leaves in $\mathcal{T}$ is denoted as $\mathcal{L}_{\mathcal{T}} \subseteq \mathcal{T}$ and its depth is denoted by $\texttt{depth}(\mathcal{T})$.

3 Multi-scale optimization (MSO)

In this section, a general framework for sequential decision-making algorithms that dynamically expand a tree on the search space to find the optimal solution, is introduced. We refer to this framework as the Multi-Scale Optimization **MSO**, as these algorithms partition the search space over multiple scales. We formally define **MSO** algorithm as follows.

Definition 2 An algorithm that constructs a *hierarchy of partitions* on the search space $\mathcal{X}$ whilst looking for f's global optimizer, is an **MSO** algorithm.

MSO algorithms differ only in their strategies of growing and using the tree further to provide a good approximation of f's global maximizer point. Hence, we introduce a generic procedure of **MSO** algorithms.

3.1 A generic procedure of MSO algorithms

An **MSO** algorithm can be regarded as a divide-and-conquer tree search algorithm. In an iterative manner: it evaluates and assesses a set of leaf nodes of its *tree on* $\mathcal{X}$; and selectively expands a subset of them.

Each node provides its approximate solution $x_{h,i} \in \mathcal{X}_{h,i}$ which is referred to as (h, i)'s representative state (or sometimes *base point*). Let $\mathcal{A}$ be an **MSO** algorithm with up to J function evaluations per node, P evaluated nodes per iteration, Q subdivided/expanded nodes per iteration, and a partition factor K. Furthermore, let us denote $\mathcal{A}$'s tree $\mathcal{T}$ after iteration $t \geq 1$ by $\mathcal{T}_t$.

Remark 1 Node expansions make the tree $\mathcal{T}$ grow with time. We may therefore index different forms of $\mathcal{T}$ by an index t to denote $\mathcal{T}$'s form $\mathcal{T}_{t+1}$ after a specific event with respect to its previous form $\mathcal{T}_t$. An event could be a single/several node(s) expansion, or a function evaluation (for which $\mathcal{T}_{t+1} = \mathcal{T}_t$). Expanding Q nodes $\{(h^q, i^q)\}_{1 \leq q \leq Q} \in \mathcal{L}_t$ at time t, where (h^q, i^q) is the qth expanded node, results in:

1. $\mathcal{T}_{t+1} = \mathcal{T}_t \cup \{(h^q + 1, i_k^q)\}_{1 \leq q \leq Q, 1 \leq k \leq K}$
2. $\mathcal{L}_{t+1} = \mathcal{L}_t \setminus \{(h^q, i^q)\}_{1 \leq q \leq Q} \cup \{(h^q + 1, i_k^q)\}_{1 \leq q \leq Q, 1 \leq k \leq K}$
3. $|\mathcal{T}_{t+1}| = |\mathcal{T}_t| + QK$

4. $|\mathcal{L}_{t+1}| = |\mathcal{L}_t| + Q(K-1)$
5. $H(\mathcal{T}_{t+1}) = \max(\texttt{depth}(\mathcal{T}_t), max_{1\le q\le Q}\, h^q + 1)$

At iteration $t+1$, two steps take place, namely *evaluation* and *expansion* illustrated as follows.[2]

1. *Leaf Node(s) Evaluation* In order to find $x_{h,i}$, $J \ge 1$ function evaluations are performed within $\mathcal{X}_{h,i}$ as a part of the sequential decision making process and out of the n-evaluation budget. These J function evaluations of a leaf node may be independent of its ancestors' and may not happen at the same iteration. We refer to the process of evaluating f within $\mathcal{X}_{h,i}$ J times as *evaluating the node* (h,i). Leaf nodes with function evaluations less than J are referred to as *under-evauated* nodes. Otherwise, they are called *evaluated* nodes. The set of evaluated nodes up to iteration t (inclusive) are denoted as $\mathcal{E}_t \subseteq \mathcal{L}_{\mathcal{T}_{t-1}}$. At each iteration, $\mathcal{A}$ **selects** $P \ge 1$ under-evaluated nodes (if any) to be evaluated. Denote the set of selected-to-be-evaluated nodes as $\mathcal{P}_{t+1} \overset{\text{def}}{=} \cup_{h\in\{0,\ldots,\texttt{depth}(\mathcal{T}_t)\}}\mathcal{P}_{t+1,h} \subseteq \mathcal{L}_{\mathcal{T}_t} \setminus \mathcal{E}_t$ where:

$$\mathcal{P}_{t+1,h} \overset{\text{def}}{=} \{(h,i) : 0 \le i \le K^h - 1, \\ (h,i) \text{is evaluable at } t+1 \text{ according to } \mathcal{A}\} \tag{8}$$

   and $|\mathcal{P}_{t+1}| \le P$.
2. *Leaf Node(s) Expansion* $\mathcal{A}$ inspects the *evaluated* leaf nodes (if any), and **selects** $Q \ge 1$ among them to be split/partitioned. These selected nodes represent the subdomain in which $\mathcal{A}$ thinks x^* potentially lies and hence a finer search is favored. We refer to the process of splitting/ partitioning a leaf node (h,i) into its K children as *expanding the node* (h,i). Denote the set of selected-to-be-expanded nodes as $\mathcal{Q}_{t+1} \overset{\text{def}}{=} \cup_{h\in\{0,\ldots,\texttt{depth}(\mathcal{T}_t)\}}\mathcal{Q}_{t+1,h} \subseteq \mathcal{E}_{t+1} \subseteq \mathcal{L}_{\mathcal{T}_t}$ where:

$$\mathcal{Q}_{t+1,h} \overset{\text{def}}{=} \{(h,i) : 0 \le i \le K^h - 1, \\ (h,i) \text{ is expandable at } t+1 \text{ according to } \mathcal{A}\} \tag{9}$$

   and $|\mathcal{Q}_{t+1}| \le Q$.

After n function evaluations, $\mathcal{A}$ returns $x(n)$:

$$x(n) \in \arg \max_{x_{h,i}:(h,i)\in\mathcal{T}} f(x_{h,i}) \tag{10}$$

as an approximate of f's global maximizer point.[3] This procedure is summarized in Algorithm 1.

One can have different **MSO** algorithms based on the defining policies of the evaluable set $\mathcal{P}$ and the expandable set $\mathcal{Q}$. In fact, an **MSO** algorithm $\mathcal{A}$ seeks a balance between two components of search [46]: *exploration*, and *exploitation*. Exploration (or global search) refers to the process of learning more about the search space. On the other hand, exploitation (or local search) is the process of acting optimally according to current knowledge. Given a finite computational budget, excessive exploration (e.g., an algorithm with a broad-search tree) leads to slow convergence, whereas excessive exploitation (e.g., an algorithm with a

[2] At $t = 1$, the root node gets evaluated J times and partitioned into K nodes; $P = Q = 1$ for all **MSO** algorithms, irrespective of their values at $t > 1$.

[3] This is the same $x(n)$ of Eq. (2).

Algorithm 1 Pseudocode for Multi-Scale Optimization

```
procedure MSO(f, 𝒳, evaluation budget)
    𝒯 ← initial tree with one node (0, 0) with
            𝒳_{0,0} = 𝒳;
    while evaluation budget is not exhausted do
        Evaluate the nodes ∈ 𝒫;
        Expand the nodes ∈ 𝒬 and
                add their child nodes in 𝒯;
    end while
    return x(n) ∈ arg max_{x_{h,i}:(h,i)∈𝒯} f(x_{h,i});
end procedure
```

deep-search tree) leads to pre-mature convergence to a local maximum and hence a trade-off must be made. Accordingly, $\mathcal{A}$ chooses $\mathcal{P}$ and $\mathcal{Q}$ to be of leaf nodes of its tree $\mathcal{T}$ that preferably possess jointly good exploration and exploitation $\mathcal{A}$-defined scores.

Let $l : \mathcal{L} \to \mathbb{R}$ be the exploitation score function and $g : \mathcal{L} \to \mathbb{R}$ be the exploration score function. The projection of a node (h, i) onto the exploitation axis is then denoted as $l_{h,i} = l((h, i))$ whereas its projection onto the exploration axis is denoted as $g_{h,i} = g((h, i))$.

Exploitation (Local) Score As the function value at (h, i)'s base point, $f(x_{h,i})$ is the best value of f within $\mathcal{X}_{h,i}$ according to $\mathcal{A}$'s current belief. It is a direct indicator of (h, i)'s exploitation (local) score. Therefore, $l_{h,i}$ is taken as $f(x_{h,i})$ or its approximate (if it is unavailable) with an absolute error less than or equal to $\eta \geq 0$:[4]

$$|l_{h,i} - f(x_{h,i})| \leq \eta \tag{11}$$

Exploration (Global) Score While $l_{h,i}$ reflects $\mathcal{A}$'s guess on the best value of f within $\mathcal{X}_{h,i}$, (h, i)'s exploration (global) score $g_{h,i}$ is regarded as the likelihood of finding a better value than $f(x_{h,i})$ within $\mathcal{X}_{h,i}$. This is correlated with the bulk of unexplored space in $\mathcal{X}_{h,i}$ and often quantified by a rough measure of its size. For example, $g_{h,i} = \texttt{depth}(\mathcal{T}) - h$ or $g_{h,i} = \sigma(\mathcal{X}_{h,i})$. Hence, one can argue that nodes of the same depth have the same exploration score or may differ up to a certain limit $\zeta \geq 0$:

$$|g_{h,i} - g_{h,j}| \leq \zeta \quad \forall h \in \mathbb{Z}_0^+,\ i, j \in \{0, \ldots, K^h - 1\}, \quad i \neq j \tag{12}$$

In the exploration-exploitation plane, each node (h, i) of $\mathcal{L}$ is represented by the point $(g_{h,i}, l_{h,i})$. Let Y be the set of these points. In other words, $Y \overset{\text{def}}{=} \{(g(x), l(x)) : x \in \mathcal{L}\}$. $\mathcal{A}$ defines a measure-of-optimality function $b : \mathbb{R}^2 \to \mathbb{R}$ such that $\mathcal{P}$ and $\mathcal{Q}$ are chosen from the set of leaf nodes whose exploration-exploitation points are the *potentially optimal set* $P(Y) \overset{\text{def}}{=} \{y \in Y : \{x \in Y : b(x) > b(y), x \neq y\} = \emptyset\}$. In other words, $P(Y)$ is the set of points $\subseteq Y$ whose level set of b (b-value) corresponds to the highest value among all points of Y. Generally, b is a weighted sum of l and g of the form:

$$b_x = b((g(x), l(x))) = l(x) + \lambda \cdot g(x), \quad x \in \mathcal{L},\ \lambda \geq 0 \tag{13}$$

trading off between local and global searches. It is important to note that g, l, b used for $\mathcal{P}$ may not be the same as for $\mathcal{Q}$.

We can visualize the process of computing these scores as projecting the leaf nodes of $\mathcal{T}$ (depicted in Fig. 1a) onto a 2-D Euclidean space $\mathbb{R}^2$, as illustrated in Fig. 1b, whose vertical

[4] Bounding the approximation error could be valid with a probability of $\gamma \geq 0$. In such case, any related analysis holds with a probability of γ.

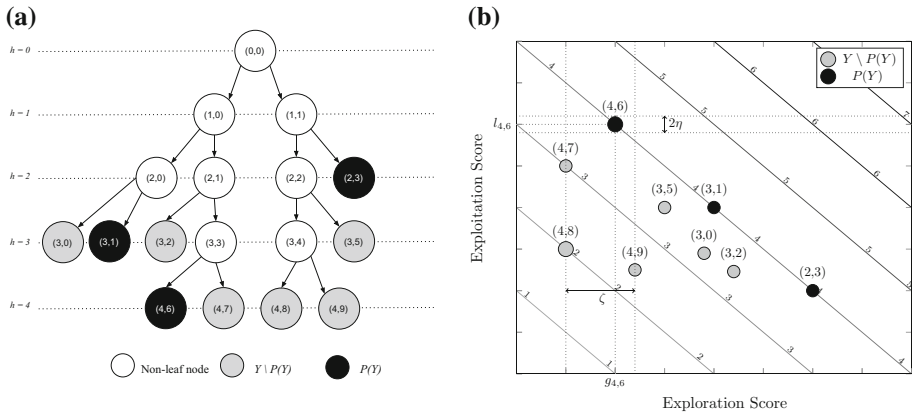

Fig. 1 The process of computing exploration and exploitation scores for the set of leaf nodes $\mathcal{L} \subseteq \mathcal{T}$ (**a**) of an **MSO** algorithm $\mathcal{A}$ can be regarded as projecting them onto a 2-D Euclidean space (**b**) $\mathbb{R}^2$ via the mapping function $s : \mathcal{L} \to \mathbb{R}^2$. Y represents $\mathcal{L}$'s image under s and $P(Y) \subseteq Y$ is the potentially optimal set. Here $P(Y)$ lies on the level set of $b(x) = l_x + \lambda \cdot g_x$ that corresponds to the value 4 (the greatest among Y's). $\mathcal{P}$ and $\mathcal{Q}$ are chosen from the set of leaf nodes whose image under s is $P(Y)$

and horizontal coordinate axes represent the domain of values for exploitation score and exploration score, respectively. From Fig. 1b, $l_{h,i}$ (11) for the node $(4, 6)$, $g_{h,i}$ (12) for nodes at depth 4, and $P(Y)$ for $\lambda = 1$ are shown.

Remark 2 One can introduce heuristics to consider, compute, and compare the b-values only for a subset of $\mathcal{L}$ at a given iteration from which $\mathcal{P}$ and $\mathcal{Q}$ are chosen rather than the whole set $\mathcal{L}$.

3.2 MSO Algorithms in the literature

In the literature, several established algorithms (such as Lipschitzian Optimization (**LO**), Dividing Rectangles (**DIRECT**), Multi-level Coordinate Search (**MCS**), Deterministic Optimistic Optimization (**DOO**), Simultaneous Optimistic Optimization (**SOO**), and their variants) satisfy the definition of **MSO** algorithms.

These algorithms can be grouped into two categories: one category requires the knowledge about f's smoothness, with the b-values being a weighted sum of the local and global scores, **LO** and **DOO** are examples of such a category; on the other hand, **DIRECT**, **MCS**, and **SOO** still make an assumption about f's smoothness, but knowledge about it may not be available. Nevertheless, these algorithms account for more than one possible setting of f's smoothness by grouping nodes based on their global scores; local scores play a role in analyzing each group separately. We describe algorithms in these two categories in accordance with the generic procedure discussed in Sect. 3.1.

*3.2.1 Lipschitzian optimization (**LO**)*

In 1972, Piyavskii [43] and Shubert [56] independently proposed Lipschitzian optimization (**LO**). It has been extended to multi-dimensional problems in [41]. **LO** assumes that f satisfies the Lipschitz condition:

$$|f(x) - f(y)| \leq L||x - y||, \ \forall x, y \in \mathcal{X} \tag{14}$$

where L, a positive real number, is the Lipschitz constant.[5] For a maximization problem, **LO** begins by evaluating the extreme points of the search space (e.g. vertices of $l \leq x \leq u$ for (1)) and constructs a piece-wise linear estimator $\hat{f}$ upper bounding f, by employing the Lipschitz condition (14). The maximum point of $\hat{f}$ represents the current estimator's best guess on where x^* lies. Furthermore, **LO** uses this point as a partitioning point of the search space into multiple subspaces on which the process of constructing an upper bound is applied recursively. In essence, **LO** iteratively refines a piece-wise linear estimator $\hat{f}$ to guide its hierarchical partitioning of the search space towards x^* where the next partitioning (sample) point is the optimum according to $\hat{f}$.

With respect to the generic procedure outlined in Sect. 3.1, **LO** has the following settings: J is of $\mathcal{O}(2^N)$; $K = P = 2^N$, that is leaf nodes created in an iteration get evaluated in the next iteration; and $Q = 1$. The two steps of **LO** at iteration $t + 1$ are summarized as the following:

1. *Leaf Node(s) Evaluation* $\mathcal{P}_{t+1}$ is $\mathcal{L}_{\mathcal{T}_t} \setminus \mathcal{E}_t$, i.e., the set of leaf nodes that are not evaluated. For a node $(h, i) \in \mathcal{P}_{t+1}$, 2^N function evaluations- at (h, i)'s vertices- are performed (hence (h, i) is $\in \mathcal{E}_{t+1}$), and the b-value ($\hat{f}(x_{h,i})$) is computed as shown in Sect. 4.2.1.
2. *Leaf Node(s) Expansion* $\mathcal{Q}_{t+1}$ is simply one node among those $\in \mathcal{E}_{t+1}$ whose b-value is the maximum, where ties are broken arbitrarily—that is, if there are more than one node whose b-value is the maximum, then any node of these is selected arbitrarily to be in $\mathcal{Q}_{t+1}$.

LO techniques are computationally complex with an exponential growth in the number of problem dimensions [25,37]. Moreover, L is often not known and overestimated resulting in a slow convergence performance. Such drawbacks limit the applicability of **LO**.

*Variants of **LO***: Techniques from combinatorial optimization [34] can be employed in **LO** constructing lower and upper bounds for $f(x_{h,i})$. Such bounds are exploited to eliminate a portion of the search space $\mathcal{X}$. In other words, some leaf nodes are *never* considered for expansion and only a small part of the tree on $\mathcal{X}$ has to be generated and processed. This results in Branch-and-Bound (**BB**)-like search algorithms [12,18,25,42]. **LO** techniques are only reliable if analytical knowledge about the function f is available, or an approximation of L is used (e.g., see [51]). For performance gains, **LO** can be parallelized as demonstrated in [14,55,62]. A general discussion on parallel algorithms for global optimization has been presented in [61].

*3.2.2 Deterministic optimistic optimization (**DOO**)*

Munos [38] proposed **DOO** by assuming f to be locally smooth (around one of its global optima) with respect to a semi-metric ℓ. This assumption in **DOO** offers a relaxation over the restrictive assumption of **LO**. **DOO** estimates, based on local smoothness, the maximum upper boud of f within a partition. Similar to **LO**, this upper bound is used to guide the hierarchical partitioning of the space.

DOO constructs a tree on $\mathcal{X}$ whose settings with respect to the generic procedure outlined in Sect. 3.1, are the following: $J = 1$; K and P are equal (as with **LO**) and treated as a

[5] In this paper, we refer to the work of Piyavskii [43] and Shubert [56] by **LO**. Nevertheless, it should be noted that Strongin [60] independently employed the Lipschitz condition for optimization problems. While Piyavskii and Shubert used a priori given constant L, Strongin proposed to adaptively estimate the Lipschitz constant during the search.

parameter by **DOO** with a default value of 3. The two steps of **DOO** at iteration $t+1$ are summarized as the following:

1. *Leaf Node(s) Evaluation* $\mathcal{P}_{t+1}$ is $\mathcal{L}_{\mathcal{T}_t} \setminus \mathcal{E}_t$, i.e., the set of leaf nodes that are not evaluated. For a node $(h,i) \in \mathcal{P}_t$, one function evaluations at (h,i)'s centre is performed (hence (h,i) becomes an evaluated node), and the b-value is computed as shown in Sect. 4.2.2.
2. *Leaf Node(s) Expansion* $\mathcal{Q}_{t+1}$ is simply one node among those $\in \mathcal{E}_{t+1}$ whose b-value is the maximum, where ties are broken arbitrarily.

DOO, similarly to **LO**, is not applicable in practice as it requires the knowledge about f's smoothness.

*3.2.3 Dividing rectangles (**DIRECT**)*

LO limitations in computational complexity and the knowledge of the Lipschitz constant L motivated Jones *et al.* to propose a search technique, **DIRECT** (stands for DIviding RECTangles) [29]. First of all, **DIRECT** does not need the knowledge of the Lipschitz constant L. Instead, it carries out the search by using all possible values of L from zero to infinity in a simultaneous framework, thereby balancing global and local search and improving the convergence speed significantly. This is captured by the heuristic rule (17). In addition to that, it introduces an additional constraint—on $\mathcal{Q}$—whose tightness depends on a parameter $\epsilon \geq 0$ of a nontrivial amount. If f_{max} is the best current function value, then $\epsilon |f_{max}|$ is the minimum amount by which the b-values of $\mathcal{Q}$ must exceed f_{max} as will be illustrated later. This ϵ-constraint (16) helps in protecting the algorithm from excessive local search. Furthermore, **DIRECT** cuts down the computational complexity from $\mathcal{O}(2^N)$ to $\mathcal{O}(1)$ by evaluating the cells' centre points (which are their base points as well) instead of their vertices.

With respect to the generic procedure outlined in Sect. 3.1, $J \leq 1$,[6] $K \leq 3N$, $P \leq 3N$, and $Q = \mathcal{O}(K^{\texttt{depth}(\mathcal{T})})$. Furthermore, let σ be a measure of a node's size such that a node (h,i) has a size of $\sigma_{h,i} = \sigma(\mathcal{X}_{h,i}) = ||d_{h,i}||_2$ where $d_{h,i}$ is $\mathcal{X}_{h,i}$'s diameter. Denote the set of evaluated-node sizes by $S_t \overset{\text{def}}{=} \{\sigma_{h,i} : (h,i) \in \mathcal{E}_t\}$; and denote the set of iteration indices $\{t, \ldots, t+|S_t|-1\}$ by I_t, where I_1 is the first iteration batch, $I_{|S_t|+1}$ is the second iteration batch, and so on. With $\acute{t} \in I_t$, $\sigma^{\acute{t}}_{S_t}$ is the $(\acute{t}-t+1)^{th}$ element of S_t in a descending order. Moreover, let f^t_{max} be the best function value achieved before t. The two steps of **DIRECT** at iteration $\acute{t} \in I_t$ are then summarized as the following:

1. *Leaf Node(s) Evaluation* $\mathcal{P}_{\acute{t}}$ is $\mathcal{L}_{\mathcal{T}_{\acute{t}-1}} \setminus \mathcal{E}_{\acute{t}-1}$, i.e., the set of leaf nodes that are not evaluated. For a node $(h,i) \in \mathcal{P}_{\acute{t}}$, one function evaluation at (h,i)'s centre is performed (hence (h,i) is $\in \mathcal{E}_{\acute{t}}$), and the b-value is computed as shown in Sect. 4.2.3.
2. *Leaf Node(s) Expansion* Let $\mathbb{Q}^{I_t}_{\acute{t}}$ be the set of evaluated nodes whose size is $\sigma^{\acute{t}}_{S_t}$; mathematically: $\mathbb{Q}^{I_t}_{\acute{t}} \overset{\text{def}}{=} \{(h,i) : (h,i) \in \mathcal{E}_t, \sigma_{h,i} = \sigma^{\acute{t}}_{S_t}\}$, and $b_{\acute{t}}$ is $\max_{(h,i)\in\mathbb{Q}^{I_t}_{\acute{t}}} b_{(h,i)}$, then $\mathcal{Q}_{\acute{t}}$ are set of nodes where each node $(h,i) \in \mathbb{Q}^{I_t}_{\acute{t}}$ such that:

$$b_{(h,i)} = b_{\acute{t}} \tag{15}$$

and there exists $\hat{L} \geq 0$ such that:

$$b_{(h,i)} + \hat{L}\sigma^{\acute{t}}_{S_t} \geq (1-\epsilon) f^t_{max} \tag{16}$$

$$b_{(h,i)} + \hat{L}\sigma^{\acute{t}}_{S_t} \geq b_{\hat{t}} + \hat{L}\sigma^{\hat{t}}_{S_t} \quad , \forall \hat{t} \in I_t \setminus \acute{t} \tag{17}$$

[6] One function evaluation may belong to one or more nodes.

If such node does not exist, the algorithm proceeds to the next iteration without expanding any node at $\acute{t}$ (see [20] for more details on how (16) and (17) are tested).

One can notice that within a batch of iterations, nodes are first contested among others of the same size, then among others of different size.

*Variants of **DIRECT*** Several interesting approaches have been proposed and inspired by **DIRECT** (see, e.g., [15,19,21,31,32,35,39,40,53,54]). While **DIRECT** used L^2-norm to compute the global score g, Gablonsky *et al.* [21] used L^∞-norm, thereby reducing the variations in the global score; and making the algorithm locally biased. Numerical experiments showed that it is good for problems with few global minima as compared to the original **DIRECT** algorithm. It has been shown in [15] that the ϵ-constraint (16) is sensitive to additive scaling and leads to a slow asymptotic convergence. Hence, in [15], a modification has been proposed by using the threshold $\epsilon|f_{max} - f_{median}|$ instead of $\epsilon|f_{max}|$ where f_{median} is the median of the observed function values. Sergeyev *et al.* [53] coupled **DIRECT**'s idea of using several values of the Lipschitz constant L instead of a unique value, with estimating a bound on f; and proposed a two-phased algorithm that is suitable for multi-modal functions. The first phase focuses on moving closer towards discovered local optima by expanding nodes whose cells are close to the current best solution. On the other hand, the second phase is oriented towards discovering new local optima by expanding nodes with high global scores and far from the current best solution.

*3.2.4 Multilevel coordinate search (**MCS**)*

Huyer and Neumaier [27] addressed the slow-convergence shortcoming of **DIRECT** in optimizing functions whose optimizers lie at the boundary of $\mathcal{X}$; and devised a global optimization algorithm (called Multilevel Coordinate Search (**MCS**)). **MCS** partitions $\mathcal{X}$ into hyperrectangles of uneven sizes whose base points are not necessarily the centre points.

For a node $(h, i) \in$ its tree $\mathcal{T}$, **MCS** assigns a rank measure $s_{h,i} \geq h$, which is used in selecting the expandable set $\mathcal{Q}$. The measure $s_{h,i}$ captures how many times a node (h, i) has been part of/candidate for an expansion process. We refer to this measure as pseudo-depth because it does not reflect the actual depth of the node. The children of node (h, i) with pseudo-depth $s_{h,i}$, can have upon creation a pseudo-depth of $s_{h,i} + 1$ or $\min(s_{h,i} + 2, s_{max})$ based on its size with respect to its siblings. The expandable set $\mathcal{Q}$ is selected based on pseudo-depth.

A node (h, i) has a set of numbers $\{n_j^{(h,i)}\}_{1\leq j\leq N}$ where n_j denotes the number of times $\mathcal{X}_{h,i}$ has been part of an expansion along coordinate j. $\mathcal{T}$'s depth is controlled through a single parameter s_{max} which forces the tree to a maximal depth of s_{max}. Given a fixed budget of node expansions, greater s_{max} reduces the probability of expanding an optimal node in $\mathcal{T}$, and hence a greater regret bound.

There are two heuristic rules employed to post-process $\mathcal{Q}$. The first rule (19) is based on $\{n_j^{(h,i)}\}_{1\leq j\leq N}$ to expand nodes which have high pseudo-depths, yet there is at least one coordinate along which their corresponding hyperrectangles have not been part of an expansion very often. The second rule (20) is to expand a node along a coordinate where the maximal expected gain in function value is large enough; the gain $\hat{e}_j^{(h,i)}$ for a node (h, i) along coordinate j is computed using a local quadratic model [27]. Accordingly, if $\max_{1\leq j\leq N} \hat{e}_j^{(h,i)}$ is large enough, (h, i) is then eligible for expansion along the coordinate $\arg\max_{1\leq j\leq N} \hat{e}_j^{(h,i)}$. If any of these rules does not hold for a node $\in \mathcal{Q}$, it is removed from $\mathcal{Q}$ and its pseudo-depth

is increased by one. Base points at depth s_{max} are put into a *shopping basket*, assuming them to be useful points. One can accelerate convergence by starting local searches from these points before putting them into the shopping basket.

With respect to the generic procedure outlined in Sect. 3.1, the settings of **MCS** are the following: $J = 1$; Based on the partitioning coordinate, P is $L_i \geq 3$ where L_i is the number of sampled points along coordinate i. Consequently, K could be $2L_i$, $2L_i - 1$, or $2L_i - 2$; and $Q = 1$. Furthermore, let I_t be the set of iteration indices $\{t, \ldots, t + s_{max} - 1\}$; I_1 is the first iteration batch, $I_{s_{max}}$ is the second iteration batch, and so on. The two steps of **MCS** at iteration $\acute{t} \in I_t$ are summarized as the following:

1. *Leaf Node(s) Evaluation* $\mathcal{P}_{\acute{t}}$ is $\mathcal{L}_{\mathcal{T}_{\acute{t}-1}} \setminus \mathcal{E}_{\acute{t}-1}$, i.e., the set of leaf nodes that are not evaluated. For a node $(h, i) \in \mathcal{P}_{\acute{t}}$, one function evaluation is performed (hence $(h, i) \in \mathcal{E}_{\acute{t}}$), and the b-value is computed as shown in Sect. 4.2.4.
2. *Leaf Node(s) Expansion* Let $\mathbb{Q}_{\acute{t}}^{I_t}$ be $\{(h, i) : (h, i) \in \mathcal{E}_{\acute{t}}, s_{h,i} = \acute{t} - t\}$ (if $\mathbb{Q}_{\acute{t}}^{I_t} = \emptyset$, the current iteration is simply skipped to $\acute{t}+1$), and $b_{\acute{t}}$ is $max_{(h,i)\in\mathbb{Q}_{\acute{t}}^{I_t}} b_{(h,i)}$ then, $\mathcal{Q}_{\acute{t}}$ is simply the node $(h, i) \in \mathbb{Q}_{\acute{t}}^{I_t}$ such that:

$$b_{(h,i)} = b_{\acute{t}} \tag{18}$$

and fulfills the one of the two heuristics:

$$\acute{t} - t > 2N \left(\min_{1 \leq j \leq N} n_j^{(h,i)} + 1 \right) \tag{19}$$

or:

$$b_{(h,i)} \geq f_{max} - \max_{1 \leq j \leq N} \hat{e}_j^{(h,i)} \tag{20}$$

where ties are broken arbitrarily. If none of the heuristic rules holds, the node's pseudo-depth is set to $s_{h,i} + 1$ and proceeds to the next iteration where it may be considered again.

*3.2.5 Simultaneous optimistic optimization (**SOO**)*

SOO [38] tries to approximate **DOO**'s behavior when ℓ is unknown. It expands simultaneously all the nodes (h, i) of its tree $\mathcal{T}$ for which there exists a semi-metric ℓ such that the corresponding upper bound would be the greatest. This is simulated by expanding at most a leaf node per depth if such node has the greatest $f(x_{h,i})$ with respect to leaf nodes of the same or lower depths. In addition to that, the algorithm takes a function $n \rightarrow h_{max}(n)$, as a parameter, which forces $\mathcal{T}$ to a maximal depth of $h_{max}(n) + 1$ after n node expansions (e.g., $h_{max}(n) = n^{\epsilon}$ where $\epsilon > 0$).

With respect to the generic procedure outlined in Sect. 3.1, the settings of **SOO** are the following: $J = 1$; K and P are equal and treated as a parameter by **SOO** with a default value of 3; and $Q = 1$. Furthermore, let I_t be the set of iteration indices $\{t, \ldots, t + h_{max}(n)\}$; I_1 is the first iteration batch, $I_{h_{max}(n)}$ is the second iteration batch, and so on. The two steps, according to [46], of **SOO** at iteration $\acute{t} \in I_t$ are summarized as the following:

1. *Leaf Node(s) Evaluation* $\mathcal{P}_{\acute{t}}$ is $\mathcal{L}_{\mathcal{T}_{\acute{t}-1}} \setminus \mathcal{E}_{\acute{t}-1}$, i.e., the set of leaf nodes that are not evaluated. For a node $(h, i) \in \mathcal{P}_{\acute{t}}$, one function evaluation at (h, i)'s centre is performed (hence (h, i) is $\in \mathcal{E}_{\acute{t}}$), and the b-value is computed as shown in Sect. 4.2.5.

2. *Leaf Node(s) Expansion* Let $\mathbb{Q}_{\acute{t}}^{I_t}$ be $\{(h,i) : (h,i) \in \mathcal{E}_{t-1}, h = \acute{t} - t\}$, and $b_{\acute{t}}$ is $\max_{(h,i)\in\mathbb{Q}_{\acute{t}}^{I_t}} b_{(h,i)}$ then $\mathcal{Q}_{\acute{t}}$ is simply the node $(h,i) \in \mathbb{Q}_{\acute{t}}^{I_t}$ such that:

$$b_{(h,i)} = b_{\acute{t}} \tag{21}$$

$$b_{(h,i)} \geq b_{\hat{t}} \ \forall \hat{t}\ t \leq \hat{t} < \acute{t} \tag{22}$$

where ties are broken arbitrarily. If no such node exists, the current iteration is simply skipped to $\acute{t} + 1$.

*Variants of **SOO***: Two main variants of **SOO** have been proposed in the literature, namely Stochastic Simultaneous Optimistic Optimization (**StoSOO**) [66], and Bayesian Multi-Scale Optimistic Optimization (**BaMSOO**) [67]. **StoSOO** addresses the situation where function evaluations are perturbed independently by noise. The b-values of **StoSOO**'s nodes are upper confidence bounds of f at their representative/base points. As a stochastic extension of **SOO**, it expands at most one node per depth after having it evaluated at its base point several times (i.e. $J \geq 1$ function evaluations and all of them are at $x_{h,i}$). Multiple evaluations per node are to ensure that the expanded nodes are with high probability close to optimal. On the other hand, **BaMSOO** was designed with the goal of cutting down the number of function evaluations incurred by **SOO**. **BaMSOO** eliminates the need for evaluating representative states $x_{h,i}$ deemed unfit by Gaussian process (GP) posterior bounds. Prior to evaluating a node (h,i), upper and lower confidence bounds (UCB, LCB) on $f(x_{h,i})$ are computed using a GP posterior fitted on the previous evaluations. These bounds are utilized to guide the search efficiently and possibly serve as the b-value of the corresponding node instead of $f(x_{h,i})$.

3.3 Discussion on MSO algorithms

A brief discussion on the expandable sets $\mathcal{Q}$ and the rules of partitioning for different **MSO** algorithms are presented below.

The Expandable Set $\mathcal{Q}$ While the selected-to-be-evaluated set $\mathcal{P}$ is almost the same for all the algorithms (a node is evaluated upon its creation), the process of selecting $\mathcal{Q}$ differs among them. Figure 2 shows a typical scenario of what $\mathcal{Q}$ could be in one (a batch of) iteration(s). Similar to Fig. 1b, it shows the evaluated leaves projected into the exploration-exploitation space. In **LO**, and **DOO**; each iteration is independent of each other; and a node is selected if it is the first node to lie on the curve from above. In **DIRECT**, **MCS**, and **SOO**; the case is different, iterations within a batch of iterations are co-dependent; a node is selected if it is among the first nodes to lie on the corresponding curve from above. However, from their visualizations, it can be argued that **SOO** and **DIRECT** have a greedier behaviour than **MCS** as they only expand a set of Pareto-optimal nodes in the exploration-exploitation plane.

Rules of Partitioning (Expanding) a Node **MSO** algorithms use different procedures for expanding (partitioning) a node into its children. On one end, **DOO**, and **SOO**, with a partitioning factor K, partition the cell/hyperrectangle of a node by a single coordinate creating K cells of equal length along that coordinate; on the other end, **LO** partitions with respect to all the N coordinates, which results in 2^N (not necessarily equal in size) cells. For **MCS**, knowledge built from sampled points is used to determine the partitioning coordinate as well as the position of the partitioning through the rules of golden section ratio and expected

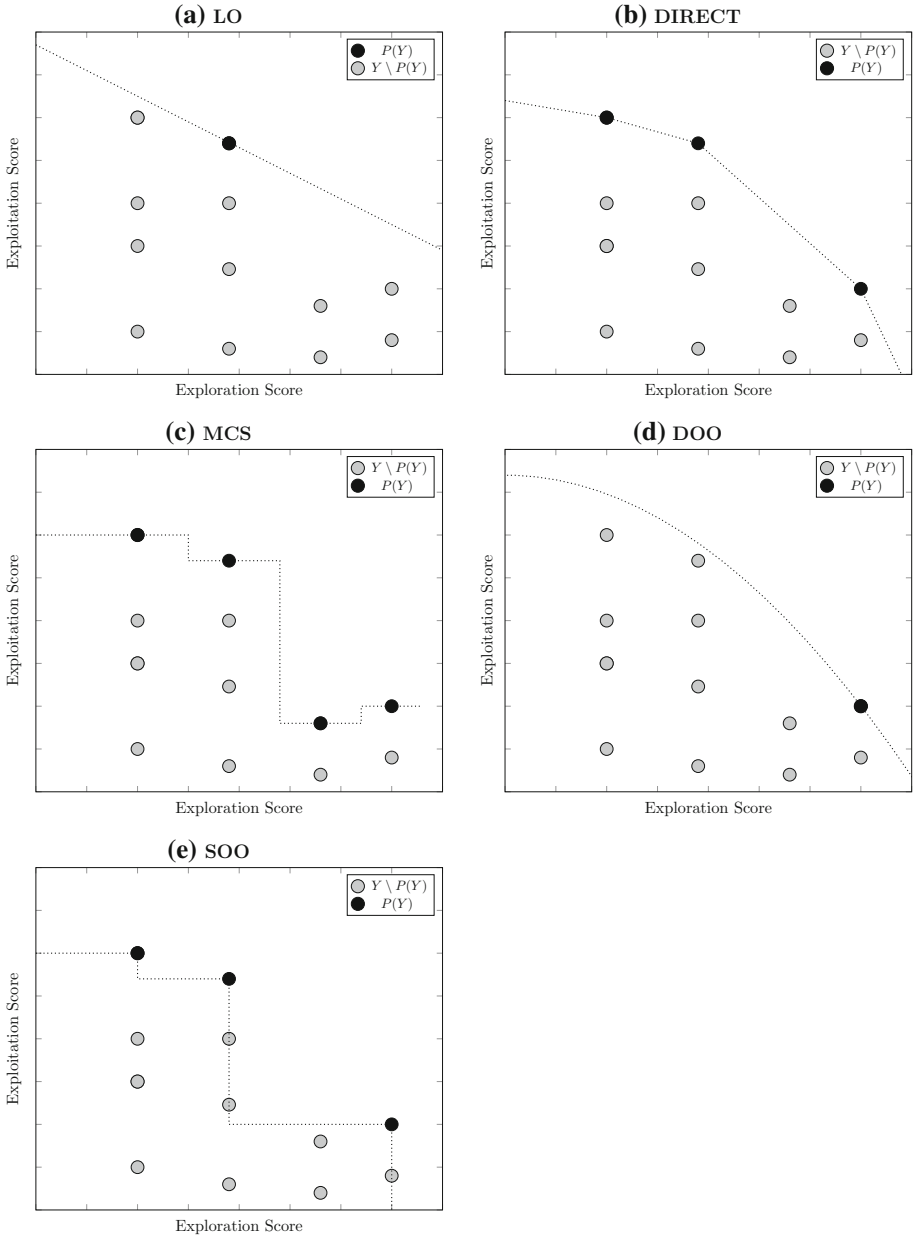

Fig. 2 Selecting expandable leaf node(s) $\mathcal{Q}$ (represented by*black dots*) for an iteration in **LO** (**a**), for a batch of iterations in **DIRECT** (**b**) , a batch of iterations in **MCS** (**c**), an iteration in **DOO** (**d**), and a batch of iterations in **SOO** (**e**). The set Y, whose elements are represented by *black* and *gray dots*, is the set of projected evaluated leaves into the exploration-exploitation space

gain [27]. **DIRECT** stands out in two aspects: its partitioning may go by a single coordinate up to N coordinates in a way that creates nodes of good local and global scores; and the fact that these partitions do not apply to the node of interest but rather to the node itself and some

of its descendants. In other words, if **DIRECT** is expanding a node (h, i) by N coordinates, the first split takes place at that node, and the m^{th} split ($m \leq N$) is applied on a node of depth $h + m - 1$ which is part of the subtree rooted at (h, i) [29]. Consequently, **DIRECT**'s tree's depth may increase by N for expanding a node in $\mathcal{Q}$.

A number of rules for partitioning a node has been investigated and numerically validated [10,30,47]. For instance, numerical experiments in [30] showed that bisecting a node's cell into two subcells works better than partitioning it into 2^N subcells using N intersecting hyperplanes.

4 Convergence analysis of MSO algorithms

In this section, we propose a theoretical methodology for finite-time analysis of **MSO** algorithms. It is then applied to analyze different **MSO** algorithms in the literature.

4.1 Theoretical framework for convergence

We derive a measure of convergence rate by analysing the complexity of the regret $r(n)$ (2). We upper-bound $r(n)$ by quantifying the amount of exploration required in each of the multi-scale partitions to achieve a near-optimal solution. In line with [38], three basic assumptions are made; the first two assumptions assist in establishing a bounded bound on $r(n)$, whereas the third assumption helps in computing it in finite time.

4.1.1 Bounding the regret $r(n)$

To bound $r(n)$ (2), one needs to assume the characteristics of f. In **LO**, f is assumed to be Lipschitz-continuous [43], whereas **DOO** and **SOO** assume local smoothness on f [38]. Here, we impose the local smoothness assumption on f, defined formally as local Hölder continuity. Let $\mathcal{T}$ be a tree on $\mathcal{X}$ created by an **MSO** algorithm and $\ell : \mathcal{X} \times \mathcal{X} \to \mathbb{R}^+$ be a semi-metric such that $\ell(x, y) = L||x - y||^\alpha$ with L and α being positive real constants.

Assumption 1 (local Hölder continuity) f is **Hölder-continuous with respect to (at least) one of its global optimizer** x^*, that is to say, $\forall x \in \mathcal{X}$, it satisfies Hölder condition [58] around x^*:

$$|f^* - f(x)| \leq L||x^* - x||^\alpha = \ell(x^*, x) \tag{23}$$

Remark 3 The class of Hölder-continuous functions is very broad. In fact, it has been shown in [20,42] that among the Lipschitz-continuous functions (which are Hölder-continuous with $\alpha = 1$) are convex/concave functions over a closed domain and continuously differentiable functions.

Before applying Assumption 1, we define the following.

Definition 3 A node $(h, i) \in \mathcal{T}$ is **optimal** if $x^* \in \mathcal{X}_{h,i}$.

Definition 4 $h_n^* \in \mathbb{Z}_0^+$ is the depth of the deepest optimal node that has been expanded up to n expansions [7] and $(h_n^*, i_n^*) \in \mathcal{T}$ is the **deepest expanded optimal node**.

[7] We are treating ***MSO*** as an n-round sequential decision making process. For ***MSO*** algorithms, n may correspond to the number of iterations, evaluations, or expansions. All three quantities are interrelated. Generally, we mean by n the number of evaluations if not stated otherwise.

Given that (h_n^*, i_n^*) is known, we can bound the regret as follows.

$$\begin{aligned} r(n) &= f^* - f(x(n)) && (24)\\ &\leq f^* - f\left(x_{h_n^*, i_n^*}\right) && (25)\\ &\leq \ell\left(x^*, x_{h_n^*, i_n^*}\right) \quad \text{from (23)} && (26)\\ &\leq sup_{x \in \mathcal{X}_{h_n^*, i_n^*}} \ell(x, x_{h_n^*, i_n^*}) && (27) \end{aligned}$$

Presume that the evaluation of a node's children is always coupled with its expansion.[8] This means that $f(x_{h_n^*+1, i_{nk}^*})$ for $1 \leq k \leq K$ are known. Consequently, there exists a tighter bound on $r(n)$ than (27) as x^* is in one of (h_n^*, i_n^*)'s children:

$$r(n) \leq sup_{1 \leq k \leq K}\ sup_{x \in \mathcal{X}_{h_n^*+1, i_{nk}^*}} \ell\left(x, x_{h_n^*+1, i_{nk}^*}\right) \leq sup_{x \in \mathcal{X}_{h_n^*, i_n^*}} \ell\left(x, x_{h_n^*, i_n^*}\right) \tag{28}$$

In order to have a bounded bound to on $r(n)$, cells of $\mathcal{T}$'s nodes need to be bounded. The next assumption implies that the bound on $r(n)$ is bounded:

Assumption 2 (Bounded Cells) There exists a decreasing sequence $\delta(h) = c\rho^h$ in h such that

$$sup_{x,y \in \mathcal{X}_{h,i}} \ell(x, y) \leq \delta(h) \ \ \forall (h, i) \in \mathcal{T} \tag{29}$$

where c is a positive real number and $\rho \in (0, 1)$.

Consequently, from (28) the regret is bounded by:

$$r(n) \leq \delta(h_n^* + 1) \tag{30}$$

Thus, finding h_n^* is the key to bound the regret. This is discussed in the next section.

4.1.2 Finding the depth of the deepest expanded optimal node, h_n^*

MSO algorithms may inevitably expand non-optimal nodes as part of their exploration -vs.- exploitation strategy. Hence, the relationship between the number of expansions n and h_n^* therefore may not be straight forward. Let us take an **MSO** algorithm $\mathcal{A}$ whose $\mathcal{T}$'s deepest expanded optimal node after n expansions is at depth $\acute{h} < \texttt{depth}(\mathcal{T})$- i.e., $h_n^* = \acute{h}$). For any node $(\acute{h}+1, i)$ to be expanded before $(\acute{h}+1, i^*)$, the optimal node at depth $\acute{h}+1$; the following must hold for $f(x_{\acute{h}+1,i})$:

$$b_{(\acute{h}+1,i)} \geq b_{(\acute{h}+1,i^*)} \tag{31}$$

$$l_{\acute{h}+1,i} + \lambda \cdot g_{\acute{h}+1,i} \geq l_{\acute{h}+1,i^*} + \lambda \cdot g_{\acute{h}+1,i^*} \tag{32}$$

from (11) and (12):

$$f(x_{\acute{h}+1,i}) + 2 \cdot \eta + \lambda \cdot \zeta \geq f\left(x_{\acute{h}+1,i^*}\right) \tag{33}$$

from (27) and (29):

$$f(x_{\acute{h}+1,i}) + 2 \cdot \eta + \lambda \cdot \zeta + \delta(\acute{h}+1) \geq f^* \tag{34}$$

Let us state three definitions to quantify and analyze nodes that satisfy (34).

[8] we shall consider this presumption throughout our analysis. In other words, for n node expansions, there are Kn node evaluations.

Definition 5 The set $\mathcal{X}_\epsilon \stackrel{\text{def}}{=} \{x \in \mathcal{X}, f(x) + \epsilon \geq f^*\}$ is the set of ϵ**-optimal states** in $\mathcal{X}$.

Definition 6 The set $\mathcal{I}_h^\epsilon \stackrel{\text{def}}{=} \{(h,i) \in \mathcal{T} : f(x_{h,i}) + \epsilon \geq f^*\}$ is the set of ϵ **-optimal nodes at depth** h in $\mathcal{T}$. For instance, $\mathcal{I}_{\acute{h}+1}^{2\cdot\eta+\lambda\cdot\zeta+\delta(\acute{h}+1)}$ is the set of nodes that satisfy (34).

Definition 7 $h_n^\epsilon \in \mathbb{Z}_0^+$ is the depth of the deepest ϵ-optimal node that has been expanded up to n expansions.

Let $\varepsilon(h) = 2 \cdot \eta + \lambda \cdot \zeta + \delta(h)$. If $\mathcal{A}$ allows expanding more than one node per depth, then we are certain that the optimal node at depth h gets expanded if all the nodes in $\mathcal{I}_h^{\varepsilon(h)}$ are expanded. Hence, h_n^* is guaranteed to be greater than or equal to h. Mathematically,

$$h_n^* \geq argmax \left\{h : h \in \{0, \ldots, \texttt{depth}(\mathcal{T}) - 1\},\ x \text{ is expanded } \forall x \in \mathcal{I}_h^{\varepsilon(h)}\right\} \tag{35}$$

Therefore, the relationship between n and h_n^* can be established by finding out how many expansions n are required to expand, at a given depth h, all the nodes in $\mathcal{I}_h^{\varepsilon(h)}$. It depends on two factors:

1. The number of the $\varepsilon(h)$-optimal nodes at depth h, $|\mathcal{I}_h^{\varepsilon(h)}|$.
2. $\mathcal{A}$'s strategy in expanding $\mathcal{I}_h^{\varepsilon(h)}$. If $\mathcal{A}$ has nodes at depth h, where $\acute{h}+1 < h < \texttt{depth}(\mathcal{T})$, then such nodes are not necessarily $\varepsilon(h)$-optimal. In other words, only a portion of the n expansions is dedicated $\cup_{h<\texttt{depth}(\mathcal{T})}\mathcal{I}_h^{\varepsilon(h)}$. This depends on $\mathcal{A}$'s expansion strategy.

While the second factor is $\mathcal{A}$-dependent, the first depends on f, ℓ, and how the nodes are shaped. The first factor is discussed in the next section. In Theorem 1 of Sect. 4.1.4, we shall demonstrate how these two factors play a role in identifying the regret bound for a class of **MSO** algorithm.

Remark 4 (Another bound for $r(n)$) The condition (34) gives us another bound on the regret:

$$r(n) \leq 2 \cdot \eta + \lambda \cdot \zeta + \delta(h)\ \forall h \in 0, \ldots, h_n^{\varepsilon(h)} \tag{36}$$

$$\leq 2 \cdot \eta + \lambda \cdot \zeta + \delta\left(h_n^{\varepsilon(h)}\right) \tag{37}$$

Note that $h_n^* \geq \hat{h}$ implies $h_n^{\varepsilon(h)} \geq \hat{h}$.

4.1.3 Bounding $|\mathcal{I}_h^{\varepsilon(h)}|$

Consider the ϵ-optimal states $\mathcal{X}_\epsilon$ in an N-dimensional space. Assume that $\acute{\ell}(x^*, x) \leq f(x^*) - f(x)$ where $\acute{\ell}(x^*, x) = \acute{L}||x^* - x||^\beta$, $\acute{L}$ and β are positive real constants.[9] From Definition 5, $\acute{\ell}(x^*, x) \leq \epsilon$, $\mathcal{X}_\epsilon$ is then an $\acute{\ell}$-hypersphere of radius ϵ centered at x^*. From Definition 6, $\mathcal{I}_h^\epsilon$ have their base points $x_{h,i}$ in $\mathcal{X}_\epsilon$. Since $x_{h,i}$ can be anywhere in $\mathcal{X}_{h,i}$, covering $\mathcal{X}_\epsilon$ by $\mathcal{I}_h^\epsilon$ is regarded as a packing problem of $\mathcal{I}_h^\epsilon$ into $\mathcal{X}_\epsilon$. Note that, in general, cells of $\mathcal{I}_h^\epsilon$ will not be spheres. A bound on $|\mathcal{I}_h^\epsilon|$ is formulated as the ratio of $\mathcal{X}_\epsilon$'s volume to the smallest volume among the cells of $\mathcal{I}_h^\epsilon$:

[9] The purpose of presenting $\acute{\ell}$ is to quantify how big $\mathcal{X}_\epsilon$ is, and consequently to introduce the near-optimality dimension (presented shortly). It does not always hold. Consider a constant function (e.g., $f(x) = 5$), for which $\acute{\ell}(x^*, x)$ should be 0. This violates the definition of $\acute{\ell}$ with $\acute{L}$ being a positive real constant. Nevertheless, it implies that $\ell(x^*, x) = 0 \leq \epsilon$, $\forall x \in \mathcal{X}$ and hence $\mathcal{X}_\epsilon = \mathcal{X}$. In other words, all the nodes at all depths have to be expanded to find the optimal node, yet each one of them is an optimal node. In summary, if $\acute{\ell}$ does not exist, the whole search space is considered as ϵ-optimal. Note that Assumption 1 implies $\beta \geq \alpha$.

$$|\mathcal{I}_h^\epsilon| \leq \frac{\sigma(\mathcal{X}_\epsilon)}{\min_{(h,i)\in\mathcal{I}_h^\epsilon} \sigma(\mathcal{X}_{h,i})} \tag{38}$$

$$= \mathcal{O}\left(\frac{\epsilon^{\frac{N}{\beta}}}{\min_{(h,i)\in\mathcal{I}_h^\epsilon} \sigma(\mathcal{X}_{h,i})}\right) \tag{39}$$

To simplify the bound further, we make an assumption about the cells shape in line with Assumption 2.

Assumption 3 [Cells Sphericity] There exists $1 \geq v > 0$ such that for any $h \in \mathbb{Z}_0^+$, any cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $v\delta(h)$.

Remark 5 The purpose of Assumption 3 is to fit a $v\delta(h)$-radius ℓ-hypersphere within cells of depth h, and hence the size of $\mathcal{I}_h^{\delta(h)}$ can be bounded more accurately. This depends on the hierarchical partitioning of the algorithm rather than the problem and holds seamlessly if the algorithm does not skew the shape of the cells. Almost all **MSO** algorithms maintain a coordinate-wise partition that does not skew the shape of their cells, and therefore, it is a reasonable assumption.

Cells Sphericity lets us have an explicit bound on the number of $\delta(h)$-optimal nodes at depth h, $|\mathcal{I}_h^{\delta(h)}|$- i.e., the number of nodes that cover the $\acute{\ell}$-hypersphere of radius $\delta(h)$ at depth h. Subsequently, $|\mathcal{I}_h^{\delta(h)}|$ has a bound of:

$$|\mathcal{I}_h^{\delta(h)}| = \mathcal{O}\left(\frac{\delta(h)^{\frac{N}{\beta}}}{\delta(h)^{\frac{N}{\alpha}}}\right) \tag{40}$$

$$= \mathcal{O}\left(\rho^{-h\left(N(\frac{1}{\alpha}-\frac{1}{\beta})\right)}\right) \tag{41}$$

which comes in one of three cases:

1. $\alpha < \beta$: At a given depth h, $|\mathcal{I}_h^{\delta(h)}|$ scales exponentially with N.
2. $\alpha = \beta$: At a given depth h, $|\mathcal{I}_h^{\delta(h)}|$ is constant and independent of N.
3. $\alpha > \beta$: This is not possible as it violates Assumption 1. Nevertheless, it tells us that if ℓ was chosen such that there is no guarantee that $f(x^*) - f(x) \leq \ell(x^*, x)$, then it is possible that $\mathcal{A}$ may expand nodes other $\delta(h)$-optimal nodes at depth h and possibly converges to a local optimum. In such case, $r(n)$ is of $\mathcal{O}(1)$.

Let d_v be $N(\frac{1}{\alpha} - \frac{1}{\beta})$. From (41), d_v can be seen as a quantitative measure of how much exploration is needed to guarantee optimality. For better understanding, Fig. 3 shows $\mathcal{X}_\epsilon$ for $N = 2$; it can be packed with $\left(\frac{\epsilon^{\frac{1}{\beta}}}{(v\epsilon)^{\frac{1}{\alpha}}}\right)^2 = v^{-\frac{2}{\alpha}}$ ℓ-circles of radius $v\epsilon$. In such case, d_v is 0.

In accordance with the Assumptions 1, 2, and 3, the next definition generalizes d_v and refers to it as the *near-optimality dimension*:

Definition 8 The **v-near-optimality dimension** is the smallest $d_v \geq 0$ such that there exists $C > 0$ and $c > 0$ such that for any $1 \geq \epsilon > 0$ and $1 \geq v > 0$, the maximal number of disjoint ℓ-hyperspheres of radius $vc\epsilon$ and contained in $\mathcal{X}_{c\epsilon}$ is less than $C\epsilon^{-d_v}$.

The next lemma reformulates and generalizes the bound of (41) in the light of the near-optimality definition.

Lemma 1 $|\mathcal{I}_h^{m\delta(h)}| \leq C\rho^{-hd_{v/m}}$, *where* $m > 0$.

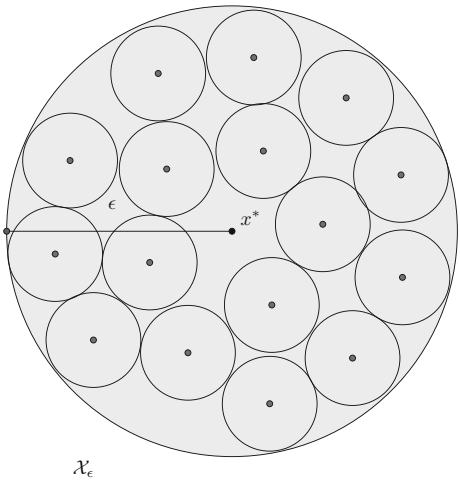

Fig. 3 $\mathcal{X}_\epsilon$ in a 2-dimensional space is an $\acute{\ell}$-circle (Here, $\acute{\ell}$ and ℓ are the l_2 norms, with α and β set to 1) centered at x^* with a radius of ϵ

Proof From Assumption 3, a cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $\frac{v}{m} \cdot m\delta(h)$. As a result, if $|\mathcal{I}_h^{m\delta(h)}|$ exceeds $C\rho^{-hd_{v/m}}$, then there are more than $C\rho^{-hd_{v/m}}$ ℓ-hypersphere of radius $\frac{v}{m} \cdot m\delta(h)$ in $\mathcal{X}_{m\delta(h)}$ which contradicts the definition of $d_{v/m}$. □

Using the above lemma, the bound of (41) can be reformulated as $|\mathcal{I}_h^{\delta(h)}| \leq C\rho^{-hd_v}$. Having constructed an explicit bound on $|\mathcal{I}_h^{\delta(h)}|$, we, now, bound $|\mathcal{I}_h^{\varepsilon(h)}|$, which is addressed in the next lemma.

Lemma 2 *For $h < h_\varepsilon$, we have $|\mathcal{I}_h^{\varepsilon(h)}| \leq C\rho^{-hd_{v/2}}$ such that*

$$h_\varepsilon \stackrel{\text{def}}{=} \min\{h : \delta(h) < 2 \cdot \eta + \lambda \cdot \zeta\} \tag{42}$$

and $d_{v/2}$ is the $v/2$-near-optimality dimension.

Proof Bounding $|\mathcal{I}_h^{\varepsilon(h)}|$ in a similar way to $|\mathcal{I}_h^{\delta(h)}|$ requires that a cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $\acute{v}(\delta(h) + 2 \cdot \eta + \lambda \cdot \zeta)$ where $1 \geq \acute{v} > 0$. From Assumption 3, we know that a cell $\mathcal{X}_{h,i}$ has an ℓ-hypersphere of radius $v\delta(h)$ where $1 \geq v > 0$. Thus, for a cell $\mathcal{X}_{h,i}$ to have an ℓ-hypersphere of radius $\acute{v}(\delta(h) + 2 \cdot \eta + \lambda \cdot \zeta)$, we need to ensure that $\acute{v}(\delta(h) + 2 \cdot \eta + \lambda \cdot \zeta) \leq v\delta(h)$. With respect to a $v/2$-near-optimality dimension, this holds for any depth h where $\delta(h) \geq 2 \cdot \eta + \lambda \cdot \zeta$ because $\frac{v}{2} \cdot 2 \cdot \delta(h) \geq \acute{v}(\delta(h) + 2 \cdot \eta + \lambda \cdot \zeta)$ where $\acute{v} = \frac{v}{2}$. Now, since $\delta(h)$ is a decreasing sequence in h, this is valid for depths less than $\min\{h : \delta(h) < 2 \cdot \eta + \lambda \cdot \zeta\}$ denoted by h_ε.

Up to this point, we know that $|\mathcal{I}_h^{\varepsilon(h)}|$ for $h < h_\varepsilon$ is upper-bounded by the maximal number of disjoint ℓ-hypersphere of $\frac{v}{2} \cdot 2 \cdot \delta(h)$ packed in $\mathcal{X}_{2\delta(h)}$. Consequently, from Definition 8 and similar to the proof of Lemma 1, we have:

$$\left|\mathcal{I}_h^{\varepsilon(h)}\right| \leq C\rho^{-hd_{v/2}}, \quad \forall h < h_\varepsilon \tag{43}$$

where $d_{v/2}$ is the $v/2$-near-optimality dimension. □

4.1.4 A convergence theorem

In this section, we present a theorem on the convergence of a class of **MSO** algorithms that adopt an expansion strategy of minimizing their trees' depths by connecting $r(n)$, h_n^*, $h_n^{\varepsilon(h)}$, and $\mathcal{I}_h^{\varepsilon(h)}$. Afterwards, three examples are worked out to see the effect of some parameters on the complexity of $r(n)$.

Theorem 1 *Let the assumptions of local Hölder continuity (Assumption 1), bounded cells (Assumption 2), and cells sphericity (Assumption 3) hold and let $\mathcal{A}$ be an **MSO** algorithm with a partition factor of K whose local and global score functions satisfy (11) and (12), respectively. Furthermore, $\mathcal{A}$ expands for each $\varepsilon(h)$-optimal node at most $M-1$ other nodes with an expansion strategy of minimizing its tree $\mathcal{T}$'s depth $\texttt{depth}(\mathcal{T})$.*[10] *Then the regret of $\mathcal{A}$ after n expansions is bounded as:*

$$r(n) \leq \delta \left(\max(\acute{h}(n), \min(h(n), h_\varepsilon)) \right) + 2 \cdot \eta \tag{44}$$

where $\acute{h}(n)$ is the smallest $h \in \mathbb{Z}_0^+$, such that:

$$M \sum_{l=0}^{\acute{h}(n)} K^l \geq n \tag{45}$$

and $h(n)$ is the smallest $h \in \mathbb{Z}_0^+$, such that:

$$CM \sum_{l=0}^{h(n)} \rho^{-l d_v/2} \geq n \tag{46}$$

and h_ε is the smallest $h \in \mathbb{Z}_0^+$ that satisfies (42).

Proof Consider any $h(n) \leq h_\varepsilon$. From the definition of $h(n)$ (46) and Lemma 2, we have:

$$M \sum_{l=0}^{h(n)-1} \left| \mathcal{I}_l^{\varepsilon(l)} \right| \leq CM \sum_{l=0}^{h(n)-1} \rho^{-l d_v/2} < n \tag{47}$$

Since for each $\varepsilon(h)$-optimal node at most M other nodes are expanded in a way that minimizes the tree's depth, we deduce from (47) that the depth of deepest expanded optimal node $h_n^* \geq h(n) - 1$ and the depth of the deepest expanded $\varepsilon(h)$-optimal node $h_n^{\varepsilon(h)} \geq h(n)$. On the other hand, for $h(n) > h_\varepsilon$, there is no valid bound on $|\mathcal{I}_l^{\varepsilon(l)}|$. Thus, $h_n^* \geq \min(h_\varepsilon, h(n)) - 1$ and $h_n^{\varepsilon(h)} \geq \min(h_\varepsilon, h(n))$. With h_n^* and $h_n^{\varepsilon(h)}$ in hand, we have two bounds on $r(n)$. Consider first the bound based on h_n^*. Let (h^*, i^*) be the child node of the deepest expanded optimal node that contains x^*, we have:

$$r(n) \leq f^* - f(h^*, i^*) \tag{48}$$

Now, since $\mathcal{A}$ deals with the approximate l (11) of f values at the nodes' base points, the regret bound, with respect to $\mathcal{A}$'s solution $l(x(n))$, is expressed as:

$$\begin{aligned} r(n) &\leq f^* - f(x(n)) + \eta \quad \text{since } |l(x(n)) - f(x(n))| \leq \eta \,, & (49)\\ &\leq f^* - f(x_{h^*,i^*}) + 2 \cdot \eta \quad \text{since } l(x(n)) \geq l(x_{h^*,i^*}) \,. & (50) \end{aligned}$$

[10] The purpose of Theorem 1 is to provide a recipe for finite-time analysis of any proposed algorithm under **MSO** framework. Nonetheless, the recipe needs to know the behavior of the algorithm. As a possible behavior, we have just assumed that the algorithm, analyzed in Theorem 1, minimizes its depth.

From Assumptions 1 & 2 and since $h^* = h_n^* + 1 \geq \min(h_\varepsilon, h(n))$, we have:

$$r(n) \leq \delta(\min(h_\varepsilon, h(n))) + 2 \cdot \eta \tag{51}$$

On the other hand, consider the bound based on $h_n^{\varepsilon(h)} \geq \min(h_\varepsilon, h(n))$. From Remark 4, the regret is bounded as $r(n) \leq 2 \cdot \eta + \lambda \cdot \zeta + \delta(\min(h_\varepsilon, h(n)))$. Clearly, the bound (51) is tighter. However, it relies on h_ε and hence can be really pessimistic (e.g. when $h_\varepsilon = 0$). We may achieve a better bound by utilizing the fact that $\mathcal{A}$'s strategy is to minimize its tree's depth; and that for n expansions, there are at least $\lfloor \frac{n}{M} \rfloor$ expanded $\varepsilon(h)$-optimal nodes. From the definition of $\acute{h}(n)$ and $\mathcal{A}$'s strategy, we deduce that $h_n^* \geq \acute{h}(n) - 1$ and $h_n^{\varepsilon(h)} \geq \acute{h}(n)$. Thus:

$$r(n) \leq \delta\left(\acute{h}(n)\right) + 2 \cdot \eta \tag{52}$$

Therefore, $r(n)$ here has two bounds, viz. (51), (52) from which we choose the tightest as in (44). □

It is important to note here that not all **MSO** algorithms aim to minimize their trees' depths. Nevertheless, the aim of the theorem is to stress that there are usually two possible approaches to obtain a regret bound: the first involves identifying the link between n and h_n^* (Sect. 4.1.2); and the second is based on identifying the link between n and $h_n^{\varepsilon(h)}$ (Remark 4). Furthermore, it showed that even when the two approaches are infeasible, an **MSO** algorithm's expansion strategy may help in establishing a better bound. The following examples evaluate the regret bound for different settings of Theorem 1's parameters.

Example 1 $\left(r(n) \text{ for } d_{v/2} = 0, \acute{h}(n) < \min(h(n), h_\varepsilon)\right)$ From (46), we have:

$$CM \sum_{l=0}^{h(n)} \rho^{-l d_{v/2}} \geq n \tag{53}$$

$$CM(h(n) + 1) \geq n \qquad \text{since } d_{v/2} = 0 \tag{54}$$

$$h(n) \geq \frac{n}{CM} - 1 \tag{55}$$

We have two cases with respect to h_ε:

1. $\eta = 0$ and $\zeta = 0$, for which $h_\varepsilon = \infty$. Therefore, $r(n)$ decays exponentially with n:

$$r(n) \leq c\rho^{\frac{n}{CM}} \tag{56}$$

2. $\eta > 0$ or $\zeta > 0$, for which $h_\varepsilon = \acute{h} < \infty$. Therefore:

$$r(n) \leq c\rho^{\min\left(\frac{n}{CM}, \acute{h}\right)} + 2 \cdot \eta \tag{57}$$

Clearly, for $1 \leq n \leq CM\acute{h}$, the regret decays exponentially with n towards $2 \cdot \eta$. For $n > CM\acute{h}$, the best bound on $r(n)$ equals $c\rho^{\acute{h}} + 2 \cdot \eta$.

Example 2 $(r(n) \text{ for } d_{v/2} > 0, \acute{h}(n) < \min(h(n), h_\varepsilon))$ From (46), we have:

$$CM \sum_{l=0}^{h(n)} \rho^{-l d_{v/2}} \geq n \tag{58}$$

From the sum of geometric series formula:

$$CM\left(\rho^{-d_{v/2}(h(n)+1)}-1\right) \geq n\left(\rho^{-d_{v/2}}-1\right) \tag{59}$$

$$\rho^{-d_{v/2}h(n)} > \frac{n}{CM}\left(1-\rho^{d_{v/2}}\right) \tag{60}$$

$$\rho^{h(n)} < \left(\frac{CM}{1-\rho^{d_{v/2}}}\right)^{\frac{1}{d_{v/2}}} \cdot n^{-\frac{1}{d_{v/2}}} \tag{61}$$

$h(n)$ is logarithmic with n. We have two cases with respect to h_ε:

1. $\eta = 0$ and $\zeta = 0$, for which $h_\varepsilon = \infty$. Therefore, $r(n)$ decays polynomially with n:

$$r(n) < c\left(\frac{CM}{1-\rho^{d_{v/2}}}\right)^{\frac{1}{d_{v/2}}} \cdot n^{-\frac{1}{d_{v/2}}} \tag{62}$$

2. $\eta > 0$ or $\zeta > 0$, for which $h_\varepsilon = \acute{h} < \infty$. Therefore:

$$r(n) \leq c \cdot \max\left(\rho^{\acute{h}}, \left(\frac{CM}{1-\rho^{d_{v/2}}}\right)^{\frac{1}{d_{v/2}}} \cdot n^{-\frac{1}{d_{v/2}}}\right) + 2 \cdot \eta \tag{63}$$

Example 3 ($r(n)$ for $\acute{h}(n) > \min(h(n), h_\varepsilon)$) From (45), we have:

$$M\sum_{l=0}^{\acute{h}(n)} K^l \geq n \tag{64}$$

$$K^{\acute{h}(n)+1} \geq \frac{n}{M}(K-1)+1 \tag{65}$$

$$\acute{h}(n) \geq \lceil \log_K(\frac{n}{M}) + \log_K(K-1) - 1 \rceil \tag{66}$$

$\acute{h}(n)$ is logarithmic with n and $r(n)$ is bounded as:

$$r(n) \leq c\rho^{\lceil \log_K(\frac{n}{M})+\log_K(K-1)-1 \rceil} + 2 * \eta \tag{67}$$

Remark 6 (**MSO** vs. Uniform Sampling) It is interesting to note that for an N-dimensional function where $|f^*-f(x)| = ||x^*-x||^\beta$, a uniform grid of Kn samples exhibits a polynomially decaying regret of $\mathcal{O}(n^{-\frac{\beta}{N}})$ whereas an **MSO** algorithm with a partition factor K and n expansions may have an exponentially decaying regret of $\mathcal{O}(\rho^{\frac{n}{CM}})$ (Example 1) irrespective of the problem dimensions N.

4.2 Analysis of MSO algorithms

Using the theoretical framework established in the previous section, we analyze the finite-time behaviour of different **MSO** algorithms in the literature.

*4.2.1 Analysis of **LO***

Several researchers have addressed the convergence analysis of **LO** techniques [24,41]. Here, we present a complimentary analysis under the framework of **MSO**. Let $N = 1$ which implies that $\mathcal{X}_{h,i}$ is an interval $[c_{h,i}, d_{h,i}]$ where $c_{h,i}$ and $d_{h,i}$ are its endpoints. Furthermore,

$\ell(x, y)$ here is $L|x - y|$. Using the Lipschitz condition (14), $\hat{f}(x_{h,i})$- and eventually $b_{(h,i)}$- is computed as:

$$b_{(h,i)} = \hat{f}(x_{h,i}) = \frac{f(c_{h,i}) + f(d_{h,i})}{2} + L\frac{d_{h,i} - c_{h,i}}{2} \tag{68}$$

This can be made equivalent to the exploitation-exploration trade-off (13) where the local score $l_{h,i} = \frac{f(c_{h,i})+f(d_{h,i})}{2}$, $\lambda = L$, and the global score $g_{h,i} = \frac{d_{h,i}-c_{h,i}}{2}$. From the Lipschitz condition (14) and Assumption 2, we have:

$$0 \leq \hat{f}(x_{h,i}) - f(x_{h,i}) \leq L(d_{h,i} - c_{h,i}) \leq \delta(h) \ . \tag{69}$$

The next lemma shows that **LO** expands $2\delta(h)$-optimal nodes in search for the optimal node at depth h.

Lemma 3 *In **LO** and at depth h, a node (h, i) is expanded before the optimal node (h^*, i^*), if:*

$$f(x_{h,i}) + 2\delta(h) \geq f^* \ . \tag{70}$$

Proof In **LO**, expanding more than one node per depth is possible; a node (h, i) is expanded before the optimal node (h^*, i^*) when $f(x_{h,i})$ satisfies the following:

$$b_{(h,i)} \geq b_{(h^*,i^*)} \tag{71}$$

$$\hat{f}(x_{h,i}) \geq \hat{f}(x_{h^*,i^*}) \tag{72}$$

From (69):

$$f(x_{h,i}) + \delta(h) \geq f(x_{h^*,i^*}) \tag{73}$$

From the Lipschitz condition (14) which is in line with Assumption 1 and Assumption 2, we have $f^* - f(x_{h^*,i^*}) \leq \sup_{x\in\mathcal{X}_{h,i}} \ell(x_{h,i}, x) \leq \delta(h)$; from which (70) follows. □

Therefore, we are certain that $h^*_n \geq h$ if all the $2\delta(h)$-optimal nodes at depth h are expanded. The following theorem establishes the regret bound for **LO** algorithms.

Theorem 2 *(Convergence of **LO**) Let $\mathcal{T}$ be **LO**'s tree on $\mathcal{X}$ and $h(n)$ be the smallest $h \in \mathbb{Z}_0^+$, such that:*

$$C\sum_{l=0}^{h(n)} \rho^{-ld_{v/2}} \geq n \tag{74}$$

*where $d_{v/2}$ is the $v/2$-near-optimality dimension and n is the number of expansions. Then the regret of **LO** is bounded as $r(n) \leq 2\delta(h(n))$.*

Proof From Lemma 3, **LO** expands only nodes that are $2\delta(h)$-optimal for $0 \leq h \leq \texttt{depth}(\mathcal{T})$. As a result, the depth of deepest expanded $2\delta(h)$-optimal node $h_n^{2\delta(h)}$ is $\texttt{depth}(\mathcal{T}) - 1$, and hence bounding the regret in the light of Remark 4 as follows:

$$r(n) \leq 2\delta(\texttt{depth}(\mathcal{T}) - 1) \tag{75}$$

Clearly, the bound depends on $\texttt{depth}(\mathcal{T})$. Let us try to make this bound more explicit by finding out the minimum $\texttt{depth}(\mathcal{T})$ with n expansions. In line with Lemma 1, we have $|\mathcal{I}_h^{2\delta(h)}| \leq C\rho^{-hd_{v/2}}$. This implies along with the definition of $h(n)$:

$$\sum_{l=0}^{h(n)-1} \left|\mathcal{I}_l^{2\delta(l)}\right| \leq C \sum_{l=0}^{h(n)-1} \rho^{-ld_{v/2}} < n \tag{76}$$

that $h_n^{2\delta(h)} \geq h(n)$. Therefore, $r(n) \leq 2\delta(h(n))$. □

Similar analysis can be applied when $N > 1$. Here, $\mathcal{X}_{h,i}$ is a hyperrectangle and $\delta(h)$ bounds a norm $L||x - y||$ rather than an interval.

4.2.2 Analysis of *DOO*

With respect to the theoretical framework, [38] provides the analysis of **DOO** for $\delta(h) < 1$. Here, we provide a generalized analysis (including $\delta(h) \geq 1$) by modifying [38, Theorem 1]. Let us start with the b-value of a node (h, i):

$$b_{(h,i)} = f(x_{h,i}) + \delta(h) \tag{77}$$

With reference to (13), $l_{h,i} = f(x_{h,i})$, $\lambda = 1$, and $g_{h,i} = \delta(h)$.

The next lemma shows that **DOO** expands $\delta(h)$-optimal nodes in search for the optimal node at depth h.

Lemma 4 *In **DOO** and at depth h, a node (h, i) is expanded before the optimal node (h^*, i^*), if:*

$$f(x_{h,i}) + \delta(h) \geq f^* \,. \tag{78}$$

Proof In **DOO**, expanding more than one node per depth is possible; a node (h, i) is expanded before the optimal node (h^*, i^*) when $f(x_{h,i})$ satisfies the following:

$$b_{(h,i)} \geq b_{(h^*,i^*)} \tag{79}$$

$$f(x_{h,i}) \geq f(x_{h^*,i^*}) \tag{80}$$

From Assumptions 1 & 2, we have $f^* - f(x_{h^*,i^*}) \leq \sup_{x \in \mathcal{X}_{h,i}} \ell(x_{h,i}, x) \leq \delta(h)$; from which (78) follows. □

Therefore, we are certain that $h_n^* \geq h$ if all the $\delta(h)$-optimal nodes at depth h are expanded. The following theorem establishes the regret bound for **DOO**.

Theorem 3 (Convergence for DOO) *Let us write $h(n)$ the smallest $h \in \mathbb{Z}_0^+$ such that*

$$C \sum_{l=0}^{h(n)} \rho^{-l d_v} \geq n \tag{81}$$

*where d_v is the v-near-optimality dimension and n is the number of expansions. Then the regret of **DOO** is bounded as $r(n) \leq \delta(h(n))$.*

Proof From Lemma 4, **DOO** expands only nodes that are $\delta(h)$-optimal for $0 \leq h \leq \texttt{depth}(\mathcal{T})$. As a result, the depth of deepest expanded $\delta(h)$-optimal node $h_n^{\delta(h)}$ is $\texttt{depth}(\mathcal{T}) - 1$, and hence bounding the regret in the light of Remark 4 as follows:

$$r(n) \leq \delta(\texttt{depth}(\mathcal{T}) - 1) \tag{82}$$

Clearly, the bound depends on $\texttt{depth}(\mathcal{T})$. Let us try to make this bound more explicit by finding out the minimum $\texttt{depth}(\mathcal{T})$ with n expansions. In line with Lemma 1, we have $|\mathcal{I}_h^{\delta(h)}| \leq C\rho^{-hd_v}$. This implies along with the definition of $h(n)$:

$$\sum_{l=0}^{h(n)-1} \left|\mathcal{I}_l^{\delta(l)}\right| \leq C \sum_{l=0}^{h(n)-1} \rho^{-l d_v} < n \tag{83}$$

that $h_n^{\delta(h)} \geq h(n)$. Therefore, $r(n) \leq \delta(h(n))$. □

4.2.3 Analysis of *DIRECT*

The b-value of a node in **DIRECT** is its local score l, with $l_{h,i} = f(x_{h,i})$, $\lambda = 0$, and $g_{h,i} = \sigma_{h,i} = ||d_{h,i}||_2$ where $d_{h,i}$ is $\mathcal{X}_{h,i}$'s diameter. As $\lambda = 0$, **DIRECT** may seem to be an exploitative **MSO** algorithm, which is not the case. The global score is employed in a heuristic (represented by (16) and (17)) for selecting $\mathcal{Q}$ within a batch of iterations, balances the exploration-vs.-exploitation trade-off. Given that the optimal node at depth $\acute{h}-1$ has been expanded, for any node $(\acute{h}, i)$ to be expanded before the optimal node $(\acute{h}, i^*)$, the following must hold (assuming (16) and (17) hold):

$$b_{(\acute{h},i)} \geq b_{(\acute{h},i^*)} \tag{84}$$

$$f(x_{\acute{h},i}) + \delta(\acute{h}) \geq f^* \tag{85}$$

For such depths, **DIRECT** expands $\delta(h)$-optimal nodes at depth h if the heuristic rules hold. However, there is no guarantee that the heuristic rules hold [15]. In [15, Theorem2], it has been shown that **DIRECT** may behave as an exhaustive grid search expanding nodes based solely on their sizes. In Theorem 4, we provide a finite-time[11] regret bound on **DIRECT** by exploiting [15]'s findings that within a batch of iteration I_t, there exists at least one node $\in \mathcal{E}_{t-1}$ that satisfies (16) and (17) and hence gets expanded within I_t. Such node is simply the node $(h_*, i_*) \in \mathcal{E}_{t-1}$ such that $b_{(h_*,i_*)} = b_t$.

Theorem 4 (Convergence of DIRECT) *Let us define $h(n)$ as is the smallest $h \in \mathbb{Z}_0^+$ such that:*

$$\sum_{l=0}^{h(n)} 3^l \geq n_I \tag{86}$$

where n_I is the greatest positive integer number such that:

$$n_I(n_I - 1) \leq \frac{2}{N^2}(n - N) \tag{87}$$

*where n is the number of expansions. Then the regret of **DIRECT** with $Q = 1$ is bounded as:*

$$r(n) \leq \delta(h(n)) \tag{88}$$

Proof **DIRECT** expands at least one node per batch of iterations; and this node is one among those of the largest size among the leaf nodes. Thus, as **DIRECT** has a partitioning factor of 3; given a number of batches n_I; and from the definition of $h(n)$, at depth $h(n) - 1$, all the nodes (optimal and non-optimal) are expanded. Hence, $r(n) \leq \delta(h(n))$.

Since we are interested in bounding the regret as a function of the number of expansions n, we need to find what is the maximum number of expansions for n_I iteration batches. In a given batch I_t, the maximum number of expansions with $Q = 1$ is $N \cdot \texttt{depth}(\mathcal{T}_{t-1})$, with $\texttt{depth}(\mathcal{T})$ growing at most by N per iteration batch. Thus, with 3 iteration batches, for instance, we can have at most $N + 1 \cdot N \cdot N + 2 \cdot N \cdot N$ expansions. For m batch of iterations, there are at most $N + N^2 \sum_{i=0}^{m-1} i = N + \frac{1}{2} \cdot N^2 \cdot m(m-1)$ expansions. Therefore, for

[11] To the best of our knowledge, there is no finite-time analysis of **DIRECT** (only the consistency property $\lim_{n\to\infty} r(n) = 0$ given by Jones *et al.* [29] which was proven again in [16] by Finkel and Kelley. Furthermore, they showed, based on nonsmooth analysis, that certain samples of **DIRECT** may converge to points that satisfy the necessary conditions for optimality defined by Clarke [8]).

n expansions, the minimum possible number of completed iteration batches is the greatest positive integer n_I iteration such that:

$$N + \frac{1}{2} \cdot N^2 \cdot n_I(n_I - 1) \leq n$$

from which (87) follows. □

4.2.4 Analysis of *MCS*

The following factors influence the analysis of **MCS**. First, the set of nodes to be considered for expansion are not of the same scale, and hence, no statement can be made about the optimality of the expanded nodes. Second, even if **MCS** is considering near-optimal nodes for expansion, the heuristics (19) and (20) may not hold which results in moving such nodes into groups of different pseudo-depth. Third, the fact that two nodes of consecutive depths h and $h+1$ may have the same size makes Assumption 2 more associated with the node's first pseudo-depth value rather than its depth h (i.e. for a node (h, i) whose s upon creation is $s_{h,i}$, then $sup_{x,y \in \mathcal{X}_{h,i}} \ell(x, y) \leq \delta(s_{h,i})$ rather than $\delta(h)$).

The b-value of a node in **MCS** is its local score, with $l_{h,i} = f(x_{h,i})$, $\lambda = 0$, and $g_{h,i} = -s_{h,i}$. The global score is used to group the nodes considered for expansion at an iteration. Assume that all the nodes keep their initial pseudo-depth; given that the optimal node at a pseudo-depth $s-1$ has been expanded, its optimal child node $(\acute{h}, i^*)$ may have a pseudo-depth of $s+1$ for which no statement can be made about the nodes with a pseudo-depth of s. Nonetheless, if $(\acute{h}, i^*)$ is at pseudo-depth s, then for any node (h, i) of the same pseudo-depth to be expanded before $(\acute{h}, i^*)$, the following must hold (assuming either of the heuristics (19) or (20) holds as well):

$$b_{(h,i)} \geq b_{(\acute{h},i^*)} \tag{89}$$

$$f(x_{h,i}) + \delta(s) \geq f^* \tag{90}$$

For such case, **MCS** expands $\delta(s)$-optimal nodes (with regards to **MCS**, we refer to a node that satisfies (90), where s is the node's initial pseudo-depth as a $\delta(s)$-optimal node, and denote the set of such nodes by $\mathcal{I}^{\delta(s)}$) and may expand non-$\delta(s)$-optimal nodes, otherwise. Clearly, the analysis is complicated. However, we can simplify it with an assumption about the structure of the maximal expected gain $\max_{1 \leq j \leq N} \hat{e}_j^{(h,i)}$ for $\delta(s)$-optimal nodes. With this assumption, the relationship between h_n^* and n can be established. This is demonstrated in the next lemma.

Lemma 5 *In **MCS**, for any depth $0 \leq h < s_{max}$, whenever $n \geq \sum_{l=0}^{s} |\mathcal{I}^{\delta(l)}| \, (s_{max} - 1)$ and $\max_{1 \leq j \leq N} \hat{e}_j^{(h,i)} \geq \delta(s)$, $\forall \delta(s)$-optimal node $\in \mathcal{T}$ where $0 \leq s \leq s_{max} - 1$, we have $h_n^* \geq s$, where n is the number of expansions.*

Proof We know that $h_n^* \geq 0$ and hence the above statement holds for $s = 0$. For $s \geq 1$, we are going to prove it by induction.

Assume that the statement holds for $0 \leq s \leq \hat{s}$. We prove it for $s \geq \hat{s} + 1$. Let $n \geq \sum_{l=0}^{\hat{s}+1} |\mathcal{I}^{\delta(l)}|(s_{max} - 1)$. Consequently, we know that $n \geq \sum_{l=0}^{\hat{s}} |\mathcal{I}^{\delta(l)}|(s_{max} - 1)$ for which $h_n^* \geq \hat{s}$. Here, we have two cases: $h_n^* \geq \hat{s} + 1$, for which the proof is done; or $h_n^* = \hat{s}$. In the second case, any node (h, i) at pseudo-depth $\hat{s} + 1$ that is expanded before the optimal node of the same pseudo-depth has to be $\delta(\hat{s} + 1)$-optimal. However, the heuristics of **MCS** may possibly cause the expansion of such nodes to be skipped and expanding at the same time non-$\delta(s)$-optimal nodes at higher pseudo-depths. Nonetheless, if the computed expected gain

for a $\delta(\hat{s}+1)$-optimal node (h,i) satisfies $\max_{1\leq j\leq N}\hat{e}_j^{(h,i)} \geq \delta(\hat{s}+1)$, then we are certain that heuristic (20) will always hold for such nodes. This can be proved as follows. Let (h,i) be a $\delta(\hat{s}+1)$-optimal node, we have:

$$f(x_{h,i}) + \delta(\hat{s}+1) \geq f^* \tag{91}$$

$$f(x_{h,i}) + \max_{1\leq j\leq N}\hat{e}_j^{(h,i)} \geq f_{max} \tag{92}$$

Thus, (20) holds for $\delta(\hat{s}+1)$-optimal nodes and they will always get expanded. Since there are $|\mathcal{I}^{\delta(\hat{s}+1)}|$ of such nodes, we are certain that the optimal node at pseudo-depth $\hat{s}+1$ is expanded after at most $|\mathcal{I}^{\delta(\hat{s}+1)}|(s_{max}-1)$ expansions. Thus, $h_n^* \geq \hat{s}+1$. □

The next theorem builds on Lemma 5 to present a finite-time analysis of **MCS** with an assumption on the structure of a node's gain. To the best of our knowledge there is no finite-time analysis of **MCS** with/without local search (only the consistency property $\lim_{n\to\infty} r(n) = 0$ for **MCS** without local search in [27]).

Theorem 5 (Convergence of MCS) *Assuming* $\max_{1\leq j\leq N}\hat{e}_j^{(h,i)} \geq \delta(s), \forall \delta(s)$*-optimal node* $\in \mathcal{T}$ *where* $0 \leq s \leq s_{max}-1$*, let us write* $s(n)$ *the smallest* $s \in \mathbb{Z}_0^+$ *such that*

$$C(s_{max}-1)\sum_{l=0}^{s(n)} \rho^{-ld_v} \geq n \tag{93}$$

where d_v *is the* v*-near-optimality dimension. Then the regret of* ***MCS*** *without local search is bounded as*

$$r(n) \leq \delta(\min(s(n), s_{max}))$$

Proof From Lemma 1, and the definition of $s(n)$ (93), we have $|\mathcal{I}^{\delta(s)}| \leq C\rho^{-sd_v}$ and:

$$\sum_{l=0}^{s(n)-1} |\mathcal{I}^{\delta(l)}|(s_{max}-1) \leq C(s_{max}-1)\sum_{l=0}^{s(n)-1} \rho^{-ld_v} < n \tag{94}$$

which implies from Lemma 5 and $\texttt{depth}(\mathcal{T}) \leq s_{max}$ that $h_n^* \geq \min(s(n)-1, s_{max}-1))$. Thus, from (30), we have $r(n) \leq \delta(\min(s(n), s_{max}))$. □

*4.2.5 Analysis of **SOO***

The b-value of a node in **SOO** is its local score, with $l_{h,i} = f(x_{h,i})$, $\lambda = 0$, and $g_{h,i} = -h$. Similar to **MCS**, the global score is used as a heuristic to filter the nodes considered for expansion at an iteration. Given that the optimal node at depth $\acute{h}-1$ has been expanded, for any node $(\acute{h}, i)$ to be expanded before the optimal node $(\acute{h}, i^*)$, the following must hold:

$$b_{(\acute{h},i)} \geq b_{(\acute{h},i^*)} \tag{95}$$

$$f(x_{\acute{h},i}) + \delta(\acute{h}) \geq f^* \tag{96}$$

For such depths, **SOO** may expand $\delta(h)$-optimal nodes . However, in contrary to **DOO**, **SOO** may expand non-$\delta(h)$-optimal nodes at depths $\acute{h} < h \leq \texttt{depth}(\mathcal{T})$. Nevertheless, the relationship between h_n^* and n can be established due to **SOO**'s strategy in sweeping $\mathcal{T}$. This is demonstrated in [38, Lemma2]. With respect to the theoretical framework, [38] provides the analysis of **SOO** for $\delta(h) < 1$. We provide a generalized analysis (including $\delta(h) \geq 1$) by modifying [38, Theorem 2]:

Theorem 6 (Convergence of SOO) *Let us write $h(n)$ the smallest $h \in \mathbb{Z}_0^+$ such that*

$$Ch_{max}(n) \sum_{l=0}^{h(n)} \rho^{-ld_v} \geq n \tag{97}$$

*where d_v is the v-near-optimality dimension and n is the number of expansions. Then the regret of **SOO** is bounded as*

$$r(n) \leq \delta(\min(h(n), h_{max} + 1))$$

Proof In line with Lemma 1, we have $|\mathcal{I}_h^{\delta(h)}| \leq C\rho^{-hd_v}$. Thus, from the definition of $h(n)$ (97) and [38, Lemma 2], the following:

$$h_{max} \sum_{l=0}^{h(n)-1} |\mathcal{I}_l^{\delta(l)}| \leq Ch_{max}(n) \sum_{l=0}^{h(n)-1} \rho^{-ld_v} < n$$

implies that $h_n^* \geq \min(h(n) - 1, h_{max}(n))$. Thus, from (30), we have $r(n) \leq \delta(\min(h(n), h_{max} + 1))$. □

Effect of $h_{max}(n)$ **SOO** controls $\mathcal{T}$'s exploration behavior (deep vs. broad) through a single parameter, namely $h_{max}(n)$. Given a fixed budget of node expansions, greater h_{max} reduces the likelihood of expanding an optimal node in $\mathcal{T}$, and hence a greater regret bound. It is interesting to consider **SOO**'s behaviour when $h_{max}(n)$ is set to ∞. Although Theorem 6 may imply that, for any n, the regret is bounded as $r(n) \leq \delta(0)$—i.e., by a constant—when $h_{max}(n) = \infty$, this is not really the case for **SOO**. When $h_{max}(n)$ is set to ∞, the regret of **SOO** is related to number of iteration batches the algorithm may have for a number of expansions n. The next corollary establishes a regret bound for **SOO** with $h_{max}(n) = \infty$. It exploits the fact that for an iteration batch I_t, the number of expansions is less than or equal $\texttt{depth}(\mathcal{T}_{t-1})$; and after each batch of iterations, $\mathcal{T}$'s depth is increased at most by one.

Corollary 1 (Convergence of SOO with $h_{max} = \infty$) *Let us define $h(n)$ as is the smallest $h \in \mathbb{Z}_0^+$ such that:*

$$C \sum_{l=0}^{h(n)} \rho^{-ld_v} \geq n_I \tag{98}$$

where n_I is the greatest positive integer number such that:

$$n_I(n_I + 1) \leq 2 \cdot n \tag{99}$$

*where n is the number of expansions. Then the regret of **SOO** with $h_{max} = \infty$ is bounded as:*

$$r(n) \leq \delta(h(n)) \tag{100}$$

Proof Let $h_n^* = \acute{h}$ after $\acute{n}$ complete batches of iteration. Then, each of the next batches expands a $\delta(\acute{h} + 1)$-optimal node (if any). Since there are $|\mathcal{I}_h^{\delta(\acute{h}+1)}| \leq C\rho^{-(\acute{h}+1)d_v}$ of such nodes, we know, after at most $\acute{n} + C\rho^{-(\acute{h}+1)d_v}$ batches, that $h_n^* \geq \acute{h} + 1$. Now, for m batches of iterations, **SOO** can have at most $\frac{m(m+1)}{2}$ expansions. Thus, from the definition of $h(n)$ and (30), we have $h_n^* \geq h(n) - 1$ and $r(n) \leq \delta(h(n))$, respectively. □

Table 1 Convergence rate of different **MSO** algorithms. These rates hold provided that Assumptions 1, 2, & 3 are satisfied

Algorithm	Convergence rate	Condition
LO	$r(n) \leq 2\delta(h(n))$	$C\sum_{l=0}^{h(n)} \rho^{-ld_v/2} \geq n$
DOO	$r(n) \leq \delta(h(n))$	$C\sum_{l=0}^{h(n)} \rho^{-ld_v} \geq n$
DIRECT	$r(n) \leq \delta(h(n))$	$\sum_{l=0}^{h(n)} 3^l \geq n_I$,
		$n_I(n_I-1) \leq \frac{2}{N^2}(n-N), Q=1$
SOO	$r(n) \leq \delta(\min(h(n), h_{max}(n)+1))$	$Ch_{max}(n)\sum_{l=0}^{h(n)} \rho^{-ld_v} \geq n$
MCS	$r(n) \leq \delta(\min(s(n), s_{max}))$	$C(s_{max}-1)\sum_{l=0}^{s(n)} \rho^{-ld_v} \geq n$,
		$\max_{1\leq j\leq N} \hat{e}_j^x \geq \delta(s), \forall x \in \mathcal{I}^{\delta(s)}\ 0 \leq s < s_{max}$

The *Condition* column provides the relation between the number of expansions n and the depth $h(n)/s(n)$ besides other algorithm-specific conditions. $h_{max}+1/s_{max}$ define the maximum depth for the trees of **SOO** and **MCS**, respectively

5 Discussion

This section presents an outline of the theoretical findings and complements it with optimization problems in practice. Table 1 summarizes the convergence rate in terms of the regret bound for the algorithms discussed. These bounds do not imply a comparative performance, but rather report their worst-case behavior. Each algorithm employs different partitioning rules for which $\delta(h)$ can be different. Nevertheless, since **LO** and **DOO** are theoretical propositions, one could comment on their comparative performance. Based on Table 1, we can conclude the following:

1. While **LO** and **DOO** assumes the knowledge about f's smoothness; for a Lipschitz-continuous function, **DOO** is more preferable than **LO** as the latter's regret bound is double the former's for the same number of expansions, provided that $d_v = d_{v/2}$, not to mention that **DOO** comes with a less restrictive assumption of the function smoothness. In practice, both algorithms are inapplicable unless some approximations on the function smoothness are made.
2. If $s_{max} = h_{max} + 1$, then **SOO** and **MCS** without local search, following the same partitioning rule, share the same regret bound, under the assumption $\max_{1\leq j\leq N} \hat{e}_j^{(h,i)} \geq \delta(s), \forall \delta(s)$-optimal node $\in \mathcal{T}$.
3. **DIRECT** has the most over-pessimistic regret bound requiring a number of expansions n that grows quadratically in the number of problem dimensions N.

5.1 A worked example

This section presents a worked example, where the loss measure bounds are computed using *Symbolic Maths* and compared with respect to the empirical results of the algorithms: **DIRECT**, **SOO**, and **MCS**.[12] Consider the function to be minimized $f(x) : \mathbb{R}^N \to \mathbb{R} = ||x - x^*||_\infty^\alpha$ over the decision space $\mathcal{X} = [-1, 1]^N$. As these algorithms evaluate their cells

[12] **LO** and **DOO** are hypothetical propositions for which only practical/adapted implementation exists.

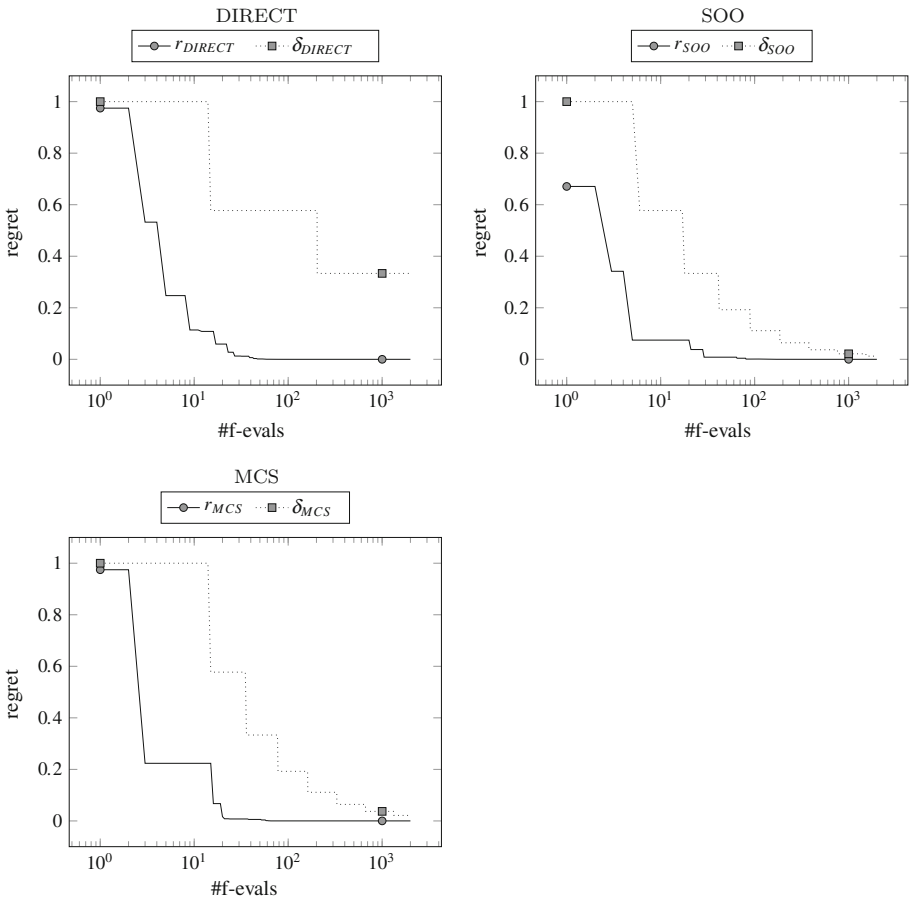

Fig. 4 The empirical convergence rate and its theoretical bound with respect to the number of function evaluations `#f-evals` for the algorithms **DIRECT**, **SOO**, and **MCS** in minimizing $||x - x^*||_\infty^\alpha$

at the center, the decreasing sequence $\delta(h)$ can be defined as $K^{-3\alpha\lfloor h/N \rfloor}$ with a partitioning factor of $K = 3$ and $v = 1/2$. As shown in Fig. 4, the bounds of **SOO** and **MCS** are tightly following their empirical regrets. On the other hand, **DIRECT**'s pessimistic behavior makes the loss bound lagging behind.

5.2 Empirical validation on real-world problems

Several studies have been conducted to evaluate the performance of **MSO** algorithms in the literature. To complement the theoretical perspective of the paper, a compilation of numerical assessment of the algorithms **DIRECT**, **SOO**, and **MCS** is presented in Figs. 5 and 6; based on the experiments in [11,44,45], using the Comparing Continuous Optimizer (COCO) methodology [22], which comes with a testbed of 24 scalable noiseless functions [23] addressing such real-world difficulties as ill-conditioning, multi-modality, and dimensionality. A set of target function values is specified per function. The algorithms are evaluated based on the number of function evalutions required to reach a target. The Expected Running Time (ERT)—used for the assessment in Figs. 5 and 6— depends on a given target function value,

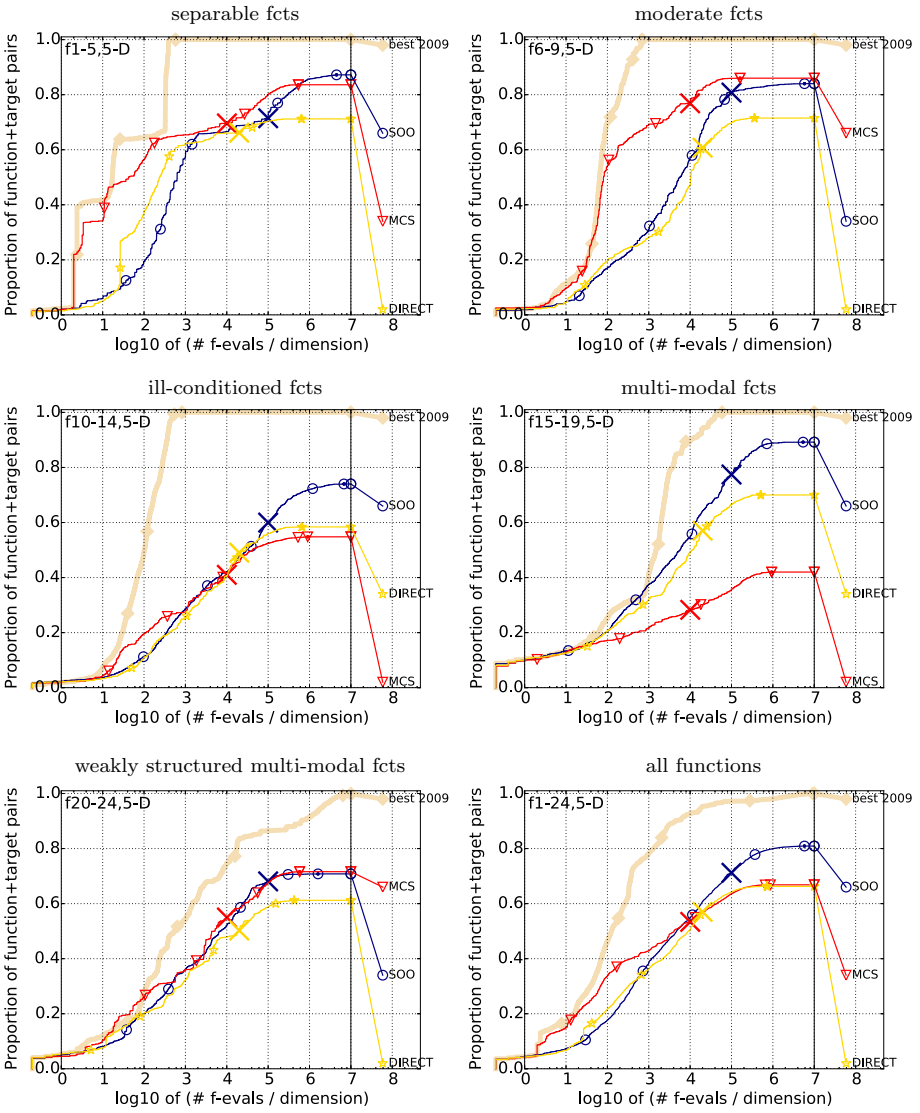

Fig. 5 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 5-D. The targets $f_{target} = f^* + \Delta f$ are defined with $\Delta f \in 10^{[-8..2]}$ such that the bestGECCO2009 artificial algorithm just not reached them within a given budget of $k \cdot DIM$, with $k \in \{0.5, 1.2, 3, 10, 50\}$. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each selected target. The cross indicates the maximum number of function evaluations used by the corresponding algorithm

$f_{target} = f^* + \Delta f$, and is computed over all relevant trials as the number of function evaluations `#f-evals` executed during each trial while the best function value did not reach f_{target} (see [22], for more details).

One can see how the pessimistic regret bound of **DIRECT** is in line with its performance as dimensionality increases: solving around 70 % (20 %) of the problems for N = 5 (N =

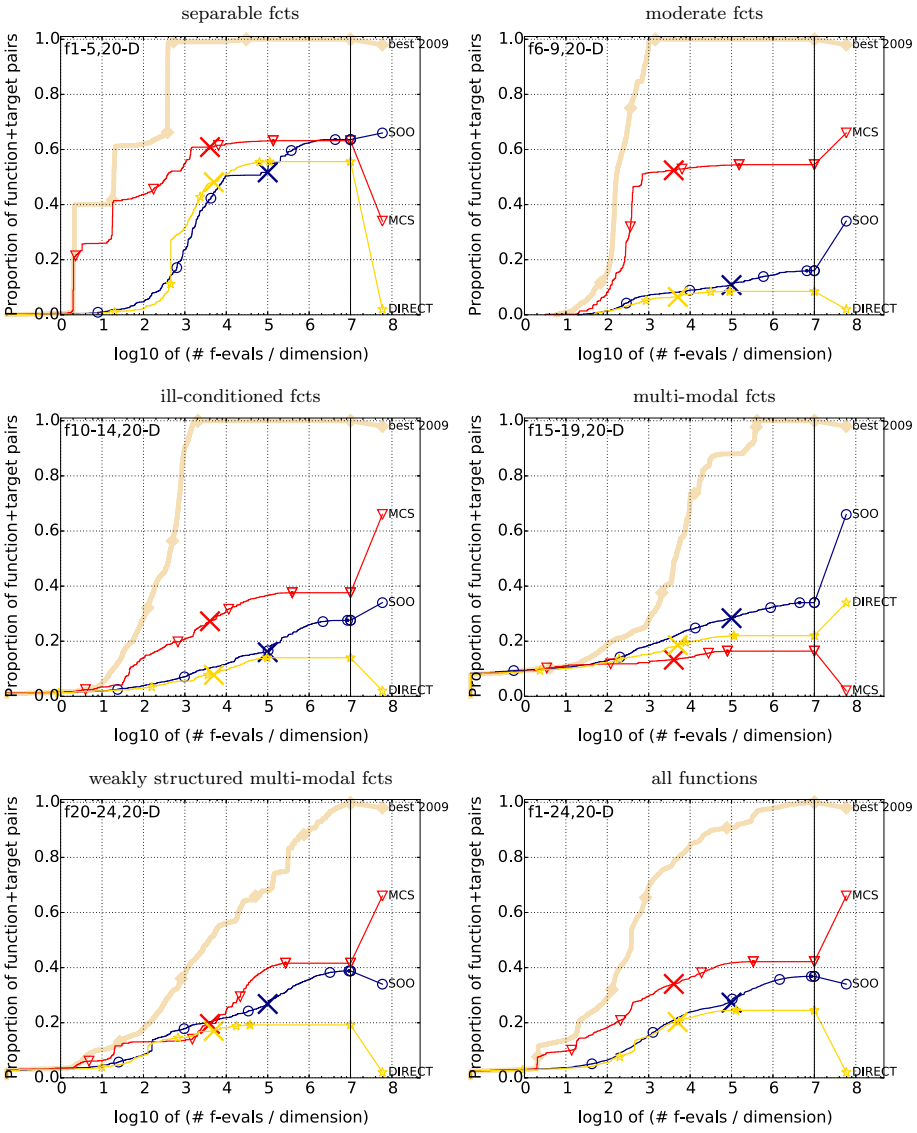

Fig. 6 Bootstrapped empirical cumulative distribution of the number of objective function evaluations divided by dimension (FEvals/DIM) for all functions and subgroups in 20-D. The targets $f_{target} = f^* + \Delta f$ are defined with $\Delta f \in 10^{[-8..2]}$ such that the bestGECCO2009 artificial algorithm just not reached them within a given budget of $k \cdot DIM$, with $k \in \{0.5, 1.2, 3, 10, 50\}$. The "best 2009" line corresponds to the best ERT observed during BBOB 2009 for each selected target. The cross indicates the maximum number of function evaluations used by the corresponding algorithm

20). Sharing similar regret bound under certain assumptions, **MCS** and **SOO** seem to have comparable performance. However, as **MCS** employs heuristic rules for expansion, it enjoys faster convergence rate, especially with higher dimensions.

6 Conclusions

This paper has consolidated a broad category of algorithms that search for the (or one) optimal solution x^* by partitioning the search space $\mathcal{X}$ over multiple scales for solving bound-constrained black-box global optimization problems, under the multi-scale optimization (**MSO**) framework. Inline with **MSO**, we present a theoretical methodology to analyze these algorithms based on three basic assumptions: a) local Hölder continuity of the objective function f, b) partitions boundedness, and c) partitions sphericity. As a result, we are able to provide a theoretical bound on the regret of several state-of-the-art **MSO** algorithms, including **LO**, **DOO**, **DIRECT**, **MCS** and **SOO**.

Acknowledgments The authors would like to thank the reviewers for their valuable comments and suggestions that had improved the paper substantially.

References

1. Archetti, F., Betrò, B.: A priori analysis of deterministic strategies for global optimization problems. Towards Global Optim. **2**, 31 (1978)
2. Auer, P., Cesa-Bianchi, N., Fischer, P.: Finite-time analysis of the multiarmed bandit problem. Mach. Learn. **47**(2–3), 235–256 (2002)
3. Bertsekas, D.P.: Constrained Optimization and Lagrange Multiplier Methods. Athena Scientific, Belmont (1996)
4. Browne, C.B., Powley, E., Whitehouse, D., Lucas, S.M., Cowling, P.I., Rohlfshagen, P., Tavener, S., Perez, D., Samothrakis, S., Colton, S.: A survey of monte carlo tree search methods. IEEE Trans. Acoust. Speech Signal Process. Comput. Intell. AI Games **4**(1), 1–43 (2012)
5. Cesa-Bianchi, N., Lugosi, G.: Prediction, Learning, and Games. Cambridge University Press, Cambridge (2006)
6. Chaput, J.C., Szostak, J.W.: Evolutionary optimization of a nonbiological atp binding protein for improved folding stability. Chem. Biol. **11**(6), 865–874 (2004)
7. Chaslot, G., Saito, J.T., Bouzy, B., Uiterwijk, J., Van Den Herik, H.J.: Monte-carlo strategies for computer go. In: Proceedings of the 18th BeNeLux Conference on Artificial Intelligence, Namur, Belgium, pp. 83–91. Citeseer (2006)
8. Clarke, F.H.: Nonsmooth analysis and optimization. In: Proceedings of the International Congress of Mathematicians (Helsinki, 1978), pp. 847–853 (1983)
9. Cousty, J., Najman, L., Perret, B.: Constructive links between some morphological hierarchies on edge-weighted graphs. In: Mathematical Morphology and Its Applications to Signal and Image Processing, pp. 86–97. Springer (2013)
10. Csendes, T., Ratz, D.: Subdivision direction selection in interval methods for global optimization. SIAM J. Numer. Anal. **34**(3), 922–938 (1997)
11. Derbel, B., Preux, P.: Simultaneous optimistic optimization on the noiseless bbob testbed. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2010–2017 (2015). doi:10.1109/CEC.2015.7257132
12. Evtushenko, Y., Posypkin, M.: A deterministic approach to global box-constrained optimization. Optim. Lett. **7**(4), 819–829 (2013)
13. Evtushenko, Y.G.: Numerical methods for finding global extrema (case of a non-uniform mesh). USSR Comput. Math. Math. Phys. **11**(6), 38–54 (1971)
14. Evtushenko, Y.G., Malkova, V., Stanevichyus, A.: Parallel global optimization of functions of several variables. Comput. Math. Math. Phys. **49**(2), 246–260 (2009)
15. Finkel, D., Kelley, C.: Additive scaling and the direct algorithm. J.Global Optim. **36**(4), 597–608 (2006)
16. Finkel, D.E., Kelley, C.T.: Convergence analysis of the direct algorithm. NCSU Mathematics Department, Raleigh, NC (2004)
17. Floudas, C.A., Pardalos, P.M., Adjiman, C., Esposito, W.R., Gümüs, Z.H., Harding, S.T., Klepeis, J.L., Meyer, C.A., Schweiger, C.A.: Handbook of Test Problems in Local and Global Optimization, vol. 33. Springer Science & Business Media, Berlin (2013)
18. Fowkes, J.M., Gould, N.I., Farmer, C.L.: A branch and bound algorithm for the global optimization of hessian lipschitz continuous functions. J. Global Optim. **56**(4), 1791–1815 (2013)

19. Gablonsky, J.: An implementation of the direct algorithm. Centre for Research in Scientific Computing, North Carolina State University, Tech. Rep. CRSC-TR98-29 (1998)
20. Gablonsky, J.: Modifications of the direct algorithm. Ph.D. thesis, North Carolina State University, Raleigh, North Carolina (2001)
21. Gablonsky, J.M., Kelley, C.T.: A locally-biased form of the direct algorithm. J. Global Optim. **21**(1), 27–37 (2001)
22. Hansen, N., Auger, A., Finck, S., Ros, R.: Real-parameter black-box optimization benchmarking 2012: Experimental setup. Tech. rep., INRIA (2012). http://coco.gforge.inria.fr/bbob2012-downloads
23. Hansen, N., Finck, S., Ros, R., Auger, A.: Real-parameter black-box optimization benchmarking 2009: Noiseless functions definitions. Tech. Rep. RR-6829, INRIA (2009). http://hal.inria.fr/inria-00362633/en/
24. Hansen, P., Jaumard, B., Lu, S.H.: On the number of iterations of piyavskii's global optimization algorithm. Math. Oper. Res. **16**(2), 334–350 (1991). doi:10.1287/moor.16.2.334
25. Horst, R., Tuy, H.: On the convergence of global methods in multiextremal optimization. J. Optim. Theory Appl. **54**(2), 253–271 (1987)
26. Hu, J., Wang, Y., Zhou, E., Fu, M.C., Marcus, S.I.: A survey of some model-based methods for global optimization. In: Optimization, Control, and Applications of Stochastic Systems, pp. 157–179. Springer (2012)
27. Huyer, W., Neumaier, A.: Global optimization by multilevel coordinate search. J. Global Optim. **14**(4), 331–355 (1999)
28. Ivanov, V.: On optimal algorithms of minimization in the class of functions with the lipschitz condition. Inf. Process. **71**, 1324–1327 (1972)
29. Jones, D.R., Perttunen, C.D., Stuckman, B.E.: Lipschitzian optimization without the lipschitz constant. J. Optim. Theory Appl. **79**(1), 157–181 (1993)
30. Kvasov, D.E., Pizzuti, C., Sergeyev, Y.D.: Local tuning and partition strategies for diagonal go methods. Numer. Math. **94**(1), 93–106 (2003)
31. Kvasov, D.E., Sergeyev, Y.D.: Lipschitz gradients for global optimization in a one-point-based partitioning scheme. J. Comput. Appl. Math. **236**(16), 4042–4054 (2012)
32. Kvasov, D.E., Sergeyev, Y.D.: Deterministic approaches for solving practical black-box global optimization problems. Adv. Eng. Softw. **80**, 58–66 (2015)
33. Lai, T.L., Robbins, H.: Asymptotically efficient adaptive allocation rules. Adv. Appl. Math. **6**(1), 4–22 (1985)
34. Laurence, A., Wolsey, G.L.N.: Integer and Combinatorial Optimization. Wiley, New York (1988)
35. Liu, Q., Cheng, W.: A modified direct algorithm with bilevel partition. J. Global Optim. **60**(3), 483–499 (2014). doi:10.1007/s10898-013-0119-1
36. Mayne, D., Polak, E.: Outer approximation algorithm for nondifferentiable optimization problems. J. Optim. Theory Appl. **42**(1), 19–30 (1984). doi:10.1007/BF00934131
37. Mladineo, F.H.: An algorithm for finding the global maximum of a multimodal, multivariate function. Math. Prog. **34**(2), 188–200 (1986). doi:10.1007/BF01580583
38. Munos, R.: Optimistic optimization of a deterministic function without the knowledge of its smoothness. In: Advances in Neural Information Processing Systems, vol. 24, pp. 783–791. Curran Associates, Inc. (2011). http://papers.nips.cc/paper/4304-optimistic-optimization-of-a-deterministic-function-without-the-knowledge-of-its-smoothness.pdf
39. Paulavičius, R., Sergeyev, Y.D., Kvasov, D.E., Žilinskas, J.: Globally-biased DISIMPL algorithm for expensive global optimization. J. Global Optim. **59**(2–3), 545–567 (2014)
40. Paulavičius, R., Žilinskas, J.: Simplicial Global Optimization. Springer, New York (2014)
41. Pintér, J.: Globally convergent methods for n-dimensional multiextremal optimization. Optimization **17**(2), 187–202 (1986)
42. Pintér, J.: Global Optimization in Action: Continuous and Lipschitz Optimization: Algorithms, Implementations and Applications, vol. 6. Springer Science & Business Media, Berlin (1995)
43. Piyavskii, S.: An algorithm for finding the absolute extremum of a function. USSR Comput. Math. Math. Phys. **12**(4), 57–67 (1972)
44. Pošík, P.: Bbob-benchmarking the direct global optimization algorithm. In: GECCO '09: Proceedings of the 11th annual conference companion on Genetic and evolutionary computation conference, pp. 2315–2320. ACM, New York, NY, USA (2009). doi:10.1145/1570256.1570323
45. Pošík, P., Huyer, W., Pál, L.: A comparison of global search algorithms for continuous black box optimization. Evolut. Comput. **20**(4), 509–541 (2012)
46. Preux, P., Munos, R., Valko, M.: Bandits attack function optimization. In: IEEE Congress on Evolutionary Computation (CEC), pp. 2245–2252 (2014)

47. Ratz, D., Csendes, T.: On the selection of subdivision directions in interval branch-and-bound methods for global optimization. J. Global Optim. **7**(2), 183–207 (1995). doi:10.1007/BF01097060
48. The Morgridge Institute for Research, I.M.: Bound constrained optimization. http://www.neos-guide.org/content/bound-constrained-optimization
49. Robbins, H., et al.: Some aspects of the sequential design of experiments. Bull. Am. Math. Soc. **58**(5), 527–535 (1952)
50. Roslund, J., Shir, O.M., Bäck, T., Rabitz, H.: Accelerated optimization and automated discovery with covariance matrix adaptation for experimental quantum control. Phys. Rev. A **80**(4), 043–415 (2009)
51. Sergeyev, Y.D.: A one-dimensional deterministic global minimization algorithm. Comput. Math. Math. Phys. **35**(5), 705–717 (1995)
52. Sergeyev, Y.D.: On convergence of divide the best global optimization algorithms. Optimization **44**(3), 303–325 (1998)
53. 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)
54. Sergeyev, Y.D., Kvasov, D.E.: A deterministic global optimization using smooth diagonal auxiliary functions. Commun. Nonlinear Scie. Numer. Simul. **21**(1), 99–111 (2015)
55. Sergeyev, Y.D., Strongin, R.G.: A global minimization algorithm with parallel iterations. USSR Comput. Math. Math. Phys. **29**(2), 7–15 (1990)
56. Shubert, B.O.: A sequential method seeking the global maximum of a function. SIAM J. Numer. Anal. **9**(3), 379–388 (1972)
57. Srinivas, N., Krause, A., Kakade, S.M., Seeger, M.: Gaussian process optimization in the bandit setting: no regret and experimental design. In: 27th International Conference on Machine Learning (2010)
58. Stover, C., Weisstein, E.W.: Hölder condition. MathWorld–A Wolfram Web Resource. http://mathworld.wolfram.com/HoelderCondition.html
59. Strongin, R.G.: Numerical methods in multi-extremal problems (information-statistical algorithms) (1978)
60. Strongin, R.G.: On the convergence of an algorithm for finding a global extremum. Eng. Cybernet. **11**, 549–555 (1973)
61. Strongin, R.G., Sergeyev, Y.: Global Optimization and Non-Convex Constraints: Sequential and Parallel Algorithms. Kluwer Academic Publishers, Dordrecht (2000)
62. Strongin, R.G., Sergeyev, Y.D.: Global multidimensional optimization on parallel computer. Parallel Comput. **18**(11), 1259–1273 (1992)
63. Sukharev, A.G.: Optimal strategies of the search for an extremum. USSR Comput. Math. Math. Phys. **11**(4), 119–137 (1971)
64. Thompson, W.R.: On the likelihood that one unknown probability exceeds another in view of the evidence of two samples. Biometrika, pp. 285–294 (1933)
65. Torn, A., Zilinskas, A.: Global Optimization. Springer, New York (1989)
66. Valko, M., Carpentier, A., Munos, R.: Stochastic simultaneous optimistic optimization. In: Proceedings of the 30th International Conference on Machine Learning (ICML-13), pp. 19–27 (2013)
67. Wang, Z., Shakibi, B., Jin, L., de Freitas, N.: Bayesian multi-scale optimistic optimization. In: Proceedings of International Conference on Artificial Intelligence and Statistics (AISTATS 2014), pp. 1005–1014 (2014)