On the use of Set Membership theory for global optimization of black-box functions

Lorenzo Sabug Jr., Fredy Ruiz, and Lorenzo Fagiano

***Abstract*— Many science and engineering applications feature non-convex optimization problems where the performance is not explicitly modeled by a cost or reward function, i.e. it is a black box. Examples include most complex design problems where experimental tests are the main method to evaluate performance of chosen values of the decision variables, in fields such as mechanics, fluid-dynamics, electromagnetics and/or magnetohydrodynamics. Solving these problems can be done iteratively: the next value of the decision variables is chosen based on the outcome generated by the previous tests. The time and resource overhead in conducting tests, however, raises the issue of most efficiently choosing the next test point according to previous observations. To approach this issue, a new global optimization strategy based on a Set Membership framework is proposed. Assuming a Lipschitz continuous cost function, the presented algorithm builds an approximation of the latter to decide whether to exploit the best result obtained so far, or to further explore the decision space. The proposed algorithm is presented and some implementation aspects are discussed. Its performance is evaluated on a set of benchmark non-convex problems and compared with those of other global optimization approaches.**

I. Introduction

In many science and engineering applications, such as mechanical design, fluid-dynamics, multi-physics simulations, control systems tuning, and chemical experiments, black-box function optimization problems arise. Black-box functions are named as such because an explicit mathematical model is unavailable, and the function values are obtained only through empirical tests, which usually entail high computational time and/or cost. Moreover, in this paper we focus on problems that may present several local minima, and where the time and resources required to carry out a test are rather large, so that the solution method shall make the most efficient use of the available trials.

Optimizing the functions and systems that fit into this description is referred to as black-box optimization [1], or derivative-free optimization [2], where only the function values are directly accessible. This is in contrast e.g. to gradient-based methods where the (first, or even further) derivatives are also assumed to be available. However these techniques usually require a rather large number of function evaluations to estimate a local quantity (i.e., the gradient at the currently evaluated point) and are designed to converge to a local optimum by always improving from the initial guess, not to explore the decision space searching for a global solution. Thus, these methods are not suitable for the problem class considered here.

Due to high overhead for function evaluations and limitations in time/resources (usually in form of a finite budget of evaluations), two conflicting aspects must be considered in choosing the next test point. *Exploitation* aims to utilize existing information about the best point and the function in general. A next sample point is then usually in the vicinity of the best known point to improve the current best value. However, because the black-box problem cannot be assumed convex, this alone can lead to local optimum. Hence, *exploration* aims to sample points to gain more information around the search space, which may have other optimal points. These aspects form the framework for most global optimization approaches. Global optimization techniques can be categorized into different classes, which include e.g. stochastic global search, smoothness-based, and response surface model-based methods.

Stochastic global search algorithms usually utilize a population of search points per iteration. According to the corresponding function values of the search points, heuristics are applied to calculate the search point locations for the next iteration (also called generation). Popular examples for this category are the particle swarm algorithm (PSO) [3] and its offshoot methods [4], [5], which have been used in various applications. However, this type of methods require numerous long function evaluations per generation, which can be much more than the limited evaluation budget.

Response surface-based methods, on the other hand, generate an approximation of the black-box function using interpolating surfaces, which maybe based on kriging [1], radial basis functions [6], or neural networks [7]. These function model approximations can then be used for the exploitation or exploration routines, whichever is appropriate.

Smoothness-based methods, especially Lipschitz-based algorithms such as DIRECT [8], [9], and AdaLIPO [10], rely on the Lipschitz constant of the function to choose the next sampling points. DIRECT uses recursive trisection of the search space into progressively smaller hyper-rectangles, to produce more accurate estimates of the optimal point. On the other hand, AdaLIPO uses a random binary variable to decide between exploitation and exploration. For exploitation, it chooses a random point within the set of potential optimizing

All authors are with Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Via Ponzio 34/5, 20133 Milan, Italy Email: {lorenzojr.sabug, fredy.ruiz, lorenzo.fagiano}@polimi.it

This research has been supported by the Italian Ministry of University and Research (MIUR) under the PRIN 2017 grant n. 201732RS94 "Systems of Tethered Multicopters"

The first author would like to acknowledge the support of the Department of Science and Technology–Science Education Institute (DOST-SEI) of the Philippines for his research

points, while for exploration it chooses a random point from the entire search space.

The present work proposes a new Lipschitz-based algorithm based on a Set Membership (SM) nonlinear function approximation [11]. The presented algorithm is a sequential optimizer which exploits the SM-based guaranteed function bounds, to choose the next test point. Using all previously acquired samples, the proposed Set Membership Optimization (SMO) updates the guaranteed lower- and upper bounds of the black-box function. The lower bounds are leveraged in the exploitation phase to choose a candidate point which is expected to exhibit a lower cost function value than the current best point. On the other hand, the difference between the SM theory-based lower- and upper bounds is treated as the uncertainty measure, and is utilized for exploration. In this case, SMO chooses a candidate point with the maximum uncertainty, to acquire more information about the function, useful to detect global optimum points throughout the search space.

This paper is organized in five sections. Section II gives the general problem statement and assumptions. Section III describes the proposed SMO algorithm. Section IV compares the performance of the proposed algorithm with other global optimization techniques for representative test functions, and Section V concludes this paper and provides insights on future directions.

II. Problem Statement

Consider a scalar function $f_o(\boldsymbol{x}),\ \boldsymbol{x} = [x_1\ x_2\ ...\ x_D]^T \in \mathcal{X}$, where $\mathcal{X}$ is a unitary hyperbox in $\mathbb{R}^D$. Without loss of generality, it is assumed that $\mathcal{X} = \{x \in \mathbb{R}^D : \|\boldsymbol{x}\|_\infty \leq 1,\ x_i \geq 0\}$.

No analytical form of the function is available. The *a priori* knowledge about f_o is given by the following assumption:

Assumption 1: f_o is Lipschitz-continuous with unknown Lipschitz constant γ_o, i.e.

$$\frac{|f_o(\boldsymbol{x}) - f_o(\boldsymbol{y})|}{\|\boldsymbol{x} - \boldsymbol{y}\|} \leq \gamma_o,\ \forall \boldsymbol{x}, \boldsymbol{y} \in \mathcal{X}$$

However, it is possible to acquire information about f_o by sampling:

Assumption 2: Given a point $\boldsymbol{x} \in \mathcal{X}$, it is possible to sample the function without noise:

$$z = f_o(\boldsymbol{x}).$$

The sampling of function f_o is assumed very expensive in terms of evaluation time and/or resources, hence it is referred to as the *long function* evaluation. Furthermore, a limited budget of N evaluations for the optimization process is given, e.g. there are only up to N long function evaluations allowed.

For any given integer $n \in \{1, \ldots, N\}$, the set of collected function samples is:

$$\boldsymbol{X}^{(n)} = \left\{(\boldsymbol{x}^{(1)},\ z^{(1)}); (\boldsymbol{x}^{(2)},\ z^{(2)}); \ldots; (\boldsymbol{x}^{(n)},\ z^{(n)})\right\}$$

The problem addressed in this paper is the search for a global optimal point of f_o using a limited amount of function samples.

Problem 1: Given a budget N of long function samples, find a minimizer $\boldsymbol{x}^*$ of the black-box function f_o:

$$\boldsymbol{x}^* \in \mathcal{X}^* = \left\{\boldsymbol{x} \mid \boldsymbol{x} = \arg\min_{\boldsymbol{x} \in \mathcal{X}} f_o(\boldsymbol{x})\right\}, \tag{1}$$

using a sequence of sample points $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots, \boldsymbol{x}^{(N)}\}$, such that $f_o(\boldsymbol{x}^*) \simeq \min z \in \{z^{(1)}, z^{(2)}, \ldots, z^{(N)}\}$.

III. Proposed Set Membership-Based Optimization Approach

The search for an optimal point $\boldsymbol{x}^*$ in Problem 1 can be approached by using a sequential procedure, as considered in [1], [8], [10]. At each iteration the next test point $\boldsymbol{x}^{(n+1)}$ is chosen given the current sampled data set $\boldsymbol{X}^{(n)}$.

In the proposed SMO algorithm, the previous data are used to derive tight bounds on the black-box function values according to the non-linear SM identification method in [11]. The resulting bounds are then leveraged for function approximation, and are utilized to choose between performing an exploitation or exploration for choosing the next test point.

A. Long function evaluation and data update

At the start of each iteration n, one long function evaluation $z^{(n)} = f_o(\boldsymbol{x}^{(n)})$ is performed at a chosen point $\boldsymbol{x}^{(n)}$, which may be the initial test point ($n = 1$), or one that was chosen at the previous iteration. This results in a new data point ($\boldsymbol{x}^{(n)},\ z^{(n)}$), which adds to the data set $\boldsymbol{X}^{(n)}$.

Due to lack of knowledge about the true Lipschitz constant γ_o, see Assumption 1, γ_o must be estimated from the collected data $\boldsymbol{X}^{(n)}$. At every iteration, an estimate $\gamma^{(n)}$ is updated given the currently available best estimate $\gamma^{(n-1)}$, and an incoming data ($\boldsymbol{x}^{(n)},\ z^{(n)}$) as:

$$\gamma^{(n)} = \max\left(\gamma^{(n-1)}, \max_{k \in [1\ \ldots\ n-1]} \frac{|z^{(n)} - z^{(k)}|}{\|\boldsymbol{x}^{(n)} - \boldsymbol{x}^{(k)}\|}\right) \tag{2}$$

Convergence of this estimate to the true Lipschitz constant as n increases has been proven, see, e.g., [12].

Given a data set $\boldsymbol{X}^{(n)}$ and $\gamma^{(n)}$ compatible with the available information on f_o, it is possible to define lower and upper bounds on f_o:

$$\underline{z}(\boldsymbol{x}) \triangleq \max_{k \in [1\ \ldots\ n]} \left(z^{(k)} - \gamma^{(n)} \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|\right), \tag{3}$$

$$\overline{z}(\boldsymbol{x}) \triangleq \min_{k \in [1\ \ldots\ n]} \left(z^{(k)} + \gamma^{(n)} \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\|\right). \tag{4}$$

These bounds represent the lowest and highest values that f_o can take in the unsampled regions of the search space. See [11] for more details. If hypotheses are true, then for any point of the search space $x \in \mathcal{X}$ it is guaranteed that $\underline{z}(\boldsymbol{x}) \leq f_o(\boldsymbol{x}) \leq \overline{z}(\boldsymbol{x})$.

A visual summary of the SM-based bounds are depicted in Fig. 1 for a one-dimensional function f_o. For higher dimensions, the functions $\underline{z}(\boldsymbol{x})$ and $\overline{z}(\boldsymbol{x})$ are intersections of hypercones, each one with vertex at a sample point. It should be emphasized that $\underline{z}$ and $\overline{z}$ are calculated from the

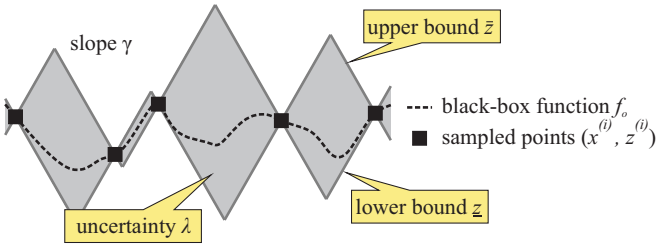

Fig. 1: Set membership-based upper and lower bounds from finite samples of f_o (1D function)

existing samples $(\boldsymbol{x}^{(k)}, z^{(k)}) \in \boldsymbol{X}^{(n)}$, and hence there is no need for additional evaluations of f_o.

Given a data set $\boldsymbol{X}^{(n)}$ the uncertainty about the value of f_o at a given point $\boldsymbol{x} \in \mathcal{X}$ is defined as:

$$\lambda(\boldsymbol{x}) = \bar{z}(\boldsymbol{x}) - \underline{z}(\boldsymbol{x}). \tag{5}$$

This function is maximized during the exploration phase of the SMO algorithm as discussed in a later subsection.

As more points are sampled especially in areas with higher λ, the bounds $\underline{z}$ and $\bar{z}$ become progressively tighter, leading to more accurate estimates of $f_o(\boldsymbol{x})$ (see Fig. 1).

B. Exploitation

In this routine, improvement of the optimal value is attempted by leveraging the current optimal point $\boldsymbol{x}^{(n)*}$ and the lower bound (hyper)cone from this location. Consider the current best point at iteration n

$$\boldsymbol{x}^{(n)*} = \boldsymbol{x}^{(r)}, r = \underset{k \in [1 \ \dots \ n]}{\arg\min} \ z^{(k)}, \tag{6}$$

which corresponds to the current best value $z^{(n)*}$. The exploitation subproblem finds a point which has the greatest potential to improve with respect to $\boldsymbol{x}^{(n)*}$. In other words, it aims to search for a point $\boldsymbol{x}_{\theta*} \in \mathcal{X}$ which gives the minimum lower bound $\underline{z}$, i.e. where f_o can go the lowest. However, as $\boldsymbol{x}$ goes farther from any sampled point $\boldsymbol{x}^{(k)}$ (see Fig. 1), uncertainty $\lambda(\boldsymbol{x})$ also increases. Hence, the search for the exploitation point $\boldsymbol{x}_{\theta*}$ is performed within the vicinity of $\boldsymbol{x}^{(n)*}$.

This search is simplified by collecting candidate points $\boldsymbol{x}_\theta^{(j)}$ for segments l_j between drawn $\boldsymbol{x}^{(n)*}$ to the other samples $\boldsymbol{x}^{(j)}$ in the set $\boldsymbol{X}^{(n)}$. The lower bounds at $\boldsymbol{x} \in l_j$ due to $\boldsymbol{x}^{(n)*}$ and $\boldsymbol{x}^{(j)}$ are defined as $z^{(n)*} - \gamma^{(n)}\|\boldsymbol{x} - \boldsymbol{x}^{(n)*}\|$ and $z^{(j)} - \gamma^{(n)}\|\boldsymbol{x} - \boldsymbol{x}^{(j)}\|$, respectively. The intersection of the lower bound (hyper)cones along the line l_j can then be derived geometrically. Let

$$d_{l_j} = \frac{z^{(j)} - z^{(n)*}}{\|\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(n)*}\|}. \tag{7}$$

Then, the intersection $\boldsymbol{x}_\theta^{(j)}$ of the lower bounds (refer to Fig. 2) along l_j is

$$\boldsymbol{x}_\theta^{(j)} = \boldsymbol{x}^{(n)*} + \frac{1 - \frac{d_{l_j}}{\gamma^{(n)}}}{2}(\boldsymbol{x}^{(j)} - \boldsymbol{x}^{(n)*}). \tag{8}$$

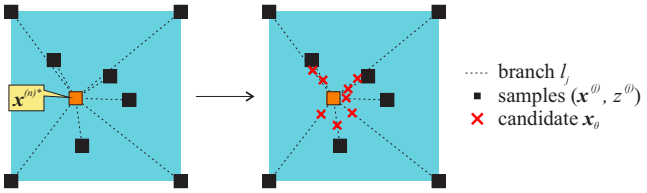

Fig. 2: Generation of exploitation candidate points $\boldsymbol{x}_\theta$ (for a 2D search space)

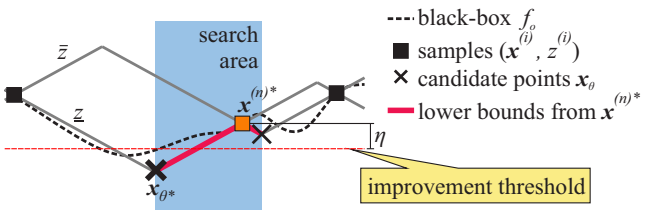

Fig. 3: Exploitation routine for a sample 1D f_o

Equation (8) is in closed form, with the simplifying assumption that there are no other intermediary hypercones which would influence $\underline{z}$ along l_j.

Consider $\boldsymbol{X}_\theta = \{\boldsymbol{x}_\theta^{(j)}\}$ as the set of candidate points. $\boldsymbol{x}_{\theta*}$ is selected as

$$\boldsymbol{x}_{\theta*} = \underset{\boldsymbol{x} \in \boldsymbol{X}_\theta}{\arg\min} \ \underline{z}(\boldsymbol{x}) \tag{9}$$

Then the corresponding lower bound $\underline{z}(\boldsymbol{x}_{\theta*})$ is understood as the best candidate improvement near to $\boldsymbol{x}^{(n)*}$. Note that the exact lower bounds $\underline{z}$ are acquired in the above $\arg\min$ operation, with the locations of $\boldsymbol{X}_\theta$ calculated in a simplified manner as (8). To evaluate if $\boldsymbol{x}_{\theta*}$ is to be taken as the next test point $\boldsymbol{x}^{(n+1)}$ for the long function evaluation, the following condition must be met

$$\underline{z}(\boldsymbol{x}_{\theta*}) \leq z^{(n)*} - \eta, \tag{10}$$

where $\eta = \alpha\sqrt{D}\gamma^{(n)}$ is referred as the *expected improvement threshold*, which considers the diameter $\sqrt{D}$ of the hypercube $\mathcal{X}$ and $\alpha \in [0, \ 1)$ is the SMO exploitation parameter (adjusted by the user). In other words, for $\boldsymbol{x}_{\theta*}$ to be acceptable, its lower bound must improve from $z^{(n)*}$ by at least η. If (10) is fulfilled (as in Fig. 3), then $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_{\theta*}$, and the long function is evaluated in the next iteration with $\boldsymbol{x}^{(n+1)}$.

However, if the previous condition is not met, SMO switches to exploration mode.

C. Exploration

The exploration phase is done in two parts: by improvement, and by uncertainty.

1) Exploration by expected improvement: Potential improvement points similar to exploitation phase are searched in entire $\mathcal{X}$, not just in the vicinity of $\boldsymbol{x}^{(n)*}$. Candidate points $\boldsymbol{x}_\phi^{(ij)}$ are generated by a mesh of branches l_{ij} between each sample pair $\{\boldsymbol{x}^{(i)}, \ \boldsymbol{x}^{(j)}\}, i \neq j$, and collected in a set $\boldsymbol{X}_\phi$. The calculation of candidate points $\boldsymbol{x}_\phi^{(ij)}$ proceeds

in a similar fashion as with exploitation using (7) and (8), but replacing $\boldsymbol{x}^{(n)*}$ with $\boldsymbol{x}^{(i)}$, and $z^{(n)*}$ with $z^{(i)}$. The best candidate point $\boldsymbol{x}_{\phi*} = \underset{\boldsymbol{x}\in\boldsymbol{X}_\phi}{\arg\min}\, \underline{z}(\boldsymbol{x})$ is selected, and subjected to the expected improvement condition (10). If this condition is fulfilled, then $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_{\phi*}$.

2) Exploration by uncertainty (discovery): If still no candidate points fulfill the expected improvement threshold, the algorithm switches to decreasing the uncertainty range λ throughout $\mathcal{X}$. In this sense, $\boldsymbol{x}^{(n+1)}$ is selected according to the associated uncertainty λ of the points in the candidate set $\boldsymbol{X}_\phi$. Sampling the point with maximum λ will of course make the uncertainty zero at that point (due to explicit evaluation), and the uncertainty around it will subsequently decrease.

Similar to exploration by improvement, the search for such a maximizer is simplified by considering the mesh connecting existing sampled points. Consider any $\boldsymbol{x}^{(i)},\ \boldsymbol{x}^{(j)} \in \boldsymbol{X}^{(n)}$. The maximum λ along the line l_{ij} between these points is at the middle, i.e. $\boldsymbol{x}_\psi^{(ij)} = \frac{\boldsymbol{x}^{(i)}+\boldsymbol{x}^{(j)}}{2}$, which is now taken as a candidate point. Hence, the set of candidates for discovery exploration is defined as

$$\boldsymbol{X}_\psi = \left\{ \boldsymbol{x}_\psi^{(ij)} \;\middle|\; \boldsymbol{x}_\psi^{(ij)} = \frac{\boldsymbol{x}^{(i)} + \boldsymbol{x}^{(j)}}{2}, i \neq j \right\}. \tag{11}$$

The point $\boldsymbol{x}_{\psi*}$ is selected such that

$$\boldsymbol{x}_{\psi*} = \underset{\boldsymbol{x}\in\boldsymbol{X}_\psi}{\arg\max}\, \lambda(\boldsymbol{x}), \tag{12}$$

and finally, $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_{\psi*}$.

D. Selection of optimal point $\boldsymbol{x}^*$

The optimal point $\boldsymbol{x}^*$ and value z^* are selected given the collected data after the iteration budget N, as follows:

$$\begin{aligned} z^* &= z^{(r)} \\ \boldsymbol{x}^* &= \boldsymbol{x}^{(r)} \\ \text{s.t. } r &= \underset{1\le k\le N}{\arg\min}\, z^{(k)}. \end{aligned} \tag{13}$$

E. Algorithm summary and implementation notes

A summarized flow of the SMO method is given as pseudo-code in Algorithm 1. A few notes are given to decrease the computational burden of the algorithm.

On the mesh generation for l_{ij}: The mesh of branches l_{ij} for the exploration routines generates $n(n-1)$ segments, which imposes exponentially increasing computational burden for each succeeding iteration. Hence, it is suggested to limit the number of branches l_{ij}, from which the candidate points are generated. For this paper, a maximum of $L = 500$ branches are generated between random pairs of sample points. There are then a maximum of 500 candidate points for the exploration routines. However, if $n(n-1) < L$, the full mesh is of course generated.

Algorithm 1: SMO Algorithm

Input: Long function f, initial point $\boldsymbol{x}^{(1)}$, iteration budget N, parameter α

```
while iteration n within the budget N do
    // Long function evaluation, data update
    Evaluate the long function f_o at x^(n) and measure output z^(n), collect sample (x^(n), z^(n)) into the set X^(n)
    Update Lipschitz constant γ^(n) and the current best sample (x^(n)*, z^(n)*) from X^(n)
    // Exploitation routine
    Use the lower bounds around the current best sample x^(n)* to choose the candidate exploitation point x_θ*
    if expected improvement condition (10) is met then
        Assign test point for next iteration x^(n+1) ← x_θ*
    else
        // Exploration by improvement
        Find x_φ* as the point from entire search space X with minimum lower bound z
        if expected improvement condition (10) is met then
            Assign test point for next iteration x^(n+1) ← x_φ*
        else
            // Exploration by uncertainty
            Find x_ψ* as the point from entire search space X with highest uncertainty λ
            Assign test point for next iteration x^(n+1) ← x_ψ*
    Go to next iteration n ← n + 1
Final optimal point and value: get the best sample (x^(n)*, z^(n)*) from the set X^(N)
```

On the coverage of search space: Due to the mechanism of mesh generation among existing sample points $\boldsymbol{X}^{(n)}$, it is noted that the algorithm only explores points inside the convex polytope generated by the existing sampled points. Hence, to ensure exploration throughout the hypercube $\mathcal{X}$, all corners $\boldsymbol{x}_\wedge$ should also be considered in the mesh generation. One approach is to first evaluate the long function f_o on all $\boldsymbol{x}_\wedge$ before actually starting with exploitation/exploration. However, this entails 2^D long function calls, which is exponential w.r.t. D.

To address this issue, a simple approach is proposed to ensure coverage in $\mathcal{X}$ without explicitly calling f_o to evaluate the corners. At the start of the algorithm, all corners are initialized with temporary value $z(\boldsymbol{x}_\wedge) = 0.0$. For each new sample $(\boldsymbol{x}^{(n)}, z^{(n)})$, the value of ith corner $z(\boldsymbol{x}_{\wedge i})$ copies the value of z of the nearest sampled point, as follows:

$$\begin{aligned} z(\boldsymbol{x}_{\wedge i}) &\leftarrow z(\boldsymbol{x}_{\wedge i*}) \\ \text{s.t. } \boldsymbol{x}_{\wedge i*} &= \boldsymbol{x}^{(r)}, r = \underset{k\in[1\ \dots\ n]}{\arg\min} \|\boldsymbol{x}^{(k)} - \boldsymbol{x}_{\wedge i}\| \end{aligned} \tag{14}$$

The concept of this workaround is shown in Fig. 4. With increasing number of sample points in $\mathcal{X}$, the corners approximate more accurately their respective values, without unnecessary calls to the long function at these points.

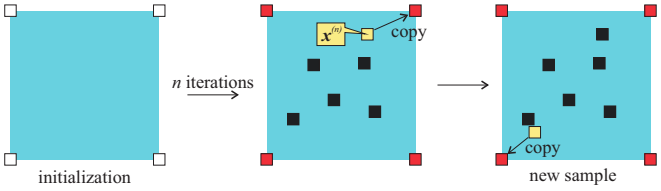

Fig. 4: Corner mirroring of nearest sample point

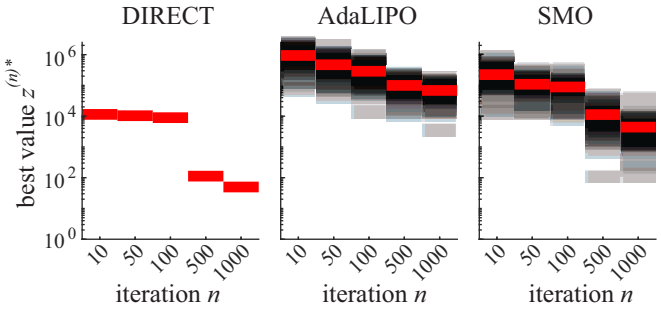

Fig. 5: Optimal value distribution w.r.t. iterations, 10D Rosenbrock function (log scale)

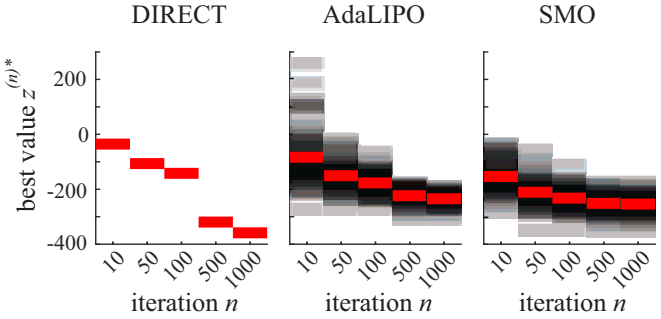

Fig. 6: Optimal value distribution w.r.t. iterations, 10D Styblinski-Tang function

IV. Performance Test Results

The proposed SMO algorithm is compared with representative Lipschitz-based methods: DIRECT and AdaLIPO.

1) Test parameters: A set of six benchmark functions of varying structures and dimensions are evaluated. Their search bounds and optimal values are summarized in Table I. The black-box functions are assumed having no available optimal value. Each algorithm is given a budget of $N = 1000$ long function evaluations in a run. The optimal results after the evaluation budget are then measured and compared among the different algorithms, in what is referred to as a fixed-cost/budget comparison [13].

To measure the statistical performance of each algorithm, 100 independent runs were performed. The starting points are different across runs, and are randomly generated such as $\boldsymbol{X}^{(1)} = \{\boldsymbol{x}_1^{(1)} \boldsymbol{x}_2^{(1)} \ \ldots \ \boldsymbol{x}_{100}^{(1)}\}$. Both AdaLIPO and SMO start at the same set of starting points $\boldsymbol{X}^{(1)}$ for these runs. On the other hand, DIRECT has intrinsically fixed search points, and does not allow random starting points. Furthermore, DIRECT does not need any parameter, while a parameter $p = 0.1$ is used for AdaLIPO and $\alpha = 0.01$ for SMO.

2) Discussion: The evolution of the optimal value $z^{(n)*}$ with respect to iteration ($n = \{10,\ 50,\ 100,\ 500,\ 1000\}$) is shown in Figs. 5, 6, and 7 for the Rosenbrock, Styblinski-Tang, and Deb's functions, respectively. In the graphs are presented the mean values (in red line), as well as the best value distribution for the 100 runs. The results of the purely deterministic DIRECT algorithm obviously does not show dispersion. On the other hand, both AdaLIPO and SMO consider randomized steps and then generate different trajectories for the best value.

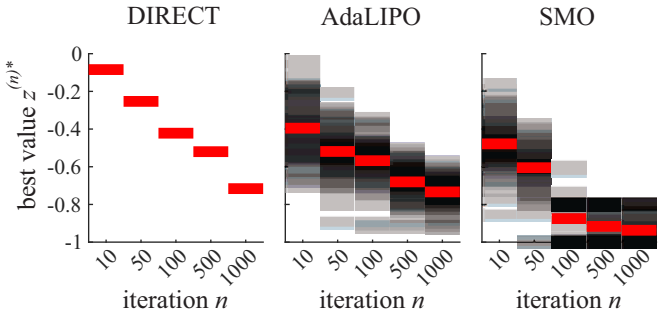

Fig. 7: Optimal value distribution w.r.t. iterations, 5D Deb's function #1

The Rosenbrock function is a standard test case for gradient-based algorithms, with a unique global minimum (unimodal) for $D = 2$; however it has 2 minima for $D = 4 \sim 30$ [14]. Due to only few minima even for $D = 10$, the recursive rectangle division by the DIRECT algorithm can be done in a relatively straightforward manner, resulting in a very low best value at the end of 1000 iterations as in Fig. 5. However, even at $n = 10$, it already achieves a very low $z^{(n)}$ compared to the other two techniques, because the initial sampling points for DIRECT were already on the low-valued regions of the search space. The DIRECT initial sampling points, which always include the very middle of the search space, contribute to its very fast convergence for parabolic or convex functions with the optimum located near the center of the search space.

The Styblinski-Tang function is similarly characterized by a few local minima, hence DIRECT can simply jump out of a particular local minimum, and exhaust all the other minima before the maximum iteration N. Algorithms with a degree of randomization such as AdaLIPO and SMO show lower performance because they do not fully exhaust the current optimal point acquired so far. However at the earlier iterations (up to 500 as in Fig. 6), both AdaLIPO and SMO display better results than DIRECT. This means that if the the iteration budget is limited, AdaLIPO and SMO would actually (on average) produce better optimal points than DIRECT.

In the case of Deb's function #1, however, SMO outperforms the other two techniques. The presence of numerous global minima is better suited for randomized and exploration-based algorithms, which can sample other regions without unnecessary exploitation in a particular area of the search space. Hence, on all stages of the optimal value evolution [Fig. 7], the AdaLIPO and SMO consistently perform better (on average) than DIRECT.

Table II summarizes the performance of the three consid-

Description	Dimensions	Function definition	Bounds	Theoretical z^*
Rosenbrock	Any D	$\sum_{i=1}^{D}[100(x_{i+1}-x_i^2)^2+(1-x_i)^2]$	$-512 \leq x_i \leq 512$	0.0
Styblinski-Tang	Any D	$f(\boldsymbol{x}) = \frac{1}{2}\sum_{i=1}^{D}\left(x_i^4 - 16x_i^2 + 5x_i\right)$	$-5 \leq x_i \leq 5$	$-39.16616D$
Deb's #1	Any D	$f(\boldsymbol{x}) = \frac{1}{D}\sum_{i=1}^{D}\sin^6(5\pi x_i)$	$-1 \leq x_i \leq 1$	-1.0
Deb's #2	Any D	$f(\boldsymbol{x}) = \frac{1}{D}\sum_{i=1}^{D}\sin^6[5\pi(x_i^{3/4} - 0.05)]$	$0 \leq x_i \leq 150$	-1.0
Schwefel	Any D	$f(\boldsymbol{x}) = -\sum_{i=1}^{D} x_i \sin\left(\sqrt{\lvert x_i \rvert}\right)$	$-500 \leq x_i \leq 500$	$-418.982D$
Salomon	Any D	$f(\boldsymbol{x}) = 1 - \cos(2\pi\sqrt{\sum_{i=1}^{D} x_i^2}) + 0.1\sqrt{\sum_{i=1}^{D} x_i^2}$	$-40 \leq x_i \leq 70$	0.0

TABLE I: Test functions used for comparative tests

Test function	D	DIRECT	AdaLIPO	SMO
Rosenbrock	10	8.69	6.21E+11 ± 3.45E+11	1.45E+08 ± 2.41E+08
Styblinski-Tang	5	-195.58	-163.82 ± 9.83	-163.33 ± 9.19
	10	-358.55	-272.90 ± 15.09	-119.05 ± 51.221
Deb's #1	5	-0.72	-0.86 ± 0.05	-0.99 ± 0.01
	10	-0.25	-0.69 ± 0.04	-0.54 ± 0.07
Deb's #2	5	-0.99	-0.86 ± 0.05	-0.98 ± 0.01
	10	-0.42	-0.70 ± 0.04	-0.93 ± 0.19
Schwefel	5	-1479.01	-1382.92 ± 123.89	-1179.22 ± 132.55
	10	-1860.45	-1946.73 ± 191.916	-1312.82 ± 128.52
Salomon	5	3.46	2.31 ± 0.47	0.62 ± 0.25
	10	4.61	5.27 ± 0.63	2.52 ± 1.26

TABLE II: Results summary for comparative tests:
average ± standard deviation of $z^{(n)*}$ after N iterations

ered algorithms in all the benchmark functions. For each test function, the algorithm producing the best optimal value is highlighted in orange and the second best in light orange. As can be observed, DIRECT produces the best results for 6 out of 11 test functions. On the other hand, SMO gives the best results for the other 5 test functions, in particular for the Deb's and Salomon functions, characterized by numerous minima. Moreover, SMO produces comparable results in the other cases except for the Rosenbrock function.

V. Conclusions and Further Work

In this work a sequential algorithm for global optimization of black-box functions has been proposed. The SMO algorithm assumes a Lipschitz-continuous cost function and is based on a nonlinear set membership function approximation. The approach is described, and implementation aspects are discussed. The selection of the test points, where the black-box function is evaluated, is carried out by solving simplified optimization problems over the guaranteed lower bound or uncertainty interval of the function, provided by a set membership model. The proposed method is tested alongside several other representative global optimization methods, comparing the quality of the best solutions after a fixed number of calls to the black-box function. The results show the competitiveness of the SMO. Convergence analysis as well as techniques to include constraints are currently under development.

References

[1] D. R. Jones, M. Schonlau, and W. J. Welch, "Efficient Global Optimization of Expensive Black-Box Functions," *Journal of Global Optimization*, vol. 13, pp. 455–492, 1998. [Online]. Available: https://link.springer.com/content/pdf/10.1023%2FA%3A1008306431147.pdf

[2] N. Pham, A. Malinowski, and T. Bartczak, "Comparative study of derivative free optimization algorithms," *IEEE Transactions on Industrial Informatics*, vol. 7, no. 4, pp. 592–600, 2011.

[3] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95 - International Conference on Neural Networks*, vol. 4. IEEE, 2014, pp. 1942–1948. [Online]. Available: http://ieeexplore.ieee.org/document/488968/

[4] Q. Liu, S. Yang, and J. Wang, "A collective neurodynamic approach to distributed constrained optimization," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 28, no. 8, pp. 1747–1758, 2017.

[5] H. Han, W. Lu, and J. Qiao, "An Adaptive Multiobjective Particle Swarm Optimization Based on Multiple Adaptive Methods," *IEEE Transactions on Cybernetics*, vol. 47, no. 9, pp. 2754–2767, 2017.

[6] M. J. Powell, "UOBYQA: Unconstrained optimization by quadratic approximation," *Mathematical Programming, Series B*, vol. 92, no. 3, pp. 555–582, 2002.

[7] Y. Wang, D. Q. Yin, S. Yang, and G. Sun, "Global and local surrogate-assisted differential evolution for expensive constrained optimization problems with inequality constraints," *IEEE Transactions on Cybernetics*, vol. 49, no. 5, pp. 1642–1656, 2019.

[8] D. R. Jones, C. D. Perttunen, and B. E. Stuckman, "Lipschitzian optimization without the Lipschitz constant," *Journal of Optimization Theory and Applications*, vol. 79, no. 1, pp. 157–181, oct 1993. [Online]. Available: http://link.springer.com/10.1007/BF00941892

[9] D. E. Finkel and C. T. Kelley, "Additive Scaling and the DIRECT Algorithm," *Journal of Global Optimization*, vol. 36, no. 4, pp. 597–608, oct 2006. [Online]. Available: http://link.springer.com/10.1007/s10898-006-9029-9

[10] C. Malherbe and N. Vayatis, "Global optimization of Lipschitz functions," *34th International Conference on Machine Learning, ICML 2017*, vol. 5, pp. 3592–3601, 2017.

[11] M. Milanese and C. Novara, "Set Membership identification of nonlinear systems," *Automatica*, vol. 40, no. 6, pp. 957–975, 2004.

[12] L. Fagiano and C. Novara, "Learning a Nonlinear Controller From Data: Theory, Computation, and Experimental Results," *IEEE Transactions on Automatic Control*, vol. 61, no. 7, pp. 1854–1868, jul 2016. [Online]. Available: http://ieeexplore.ieee.org/document/7271025/

[13] V. Beiranvand, W. Hare, and Y. Lucet, "Best practices for comparing optimization algorithms," *Optimization and Engineering*, vol. 18, no. 4, pp. 815–848, sep 2017. [Online]. Available: http://arxiv.org/abs/1709.08242 http://dx.doi.org/10.1007/s11081-017-9366-1

[14] Y.-W. Shang and Y.-H. Qiu, "A Note on the Extended Rosenbrock Function," *Evolutionary Computation*, vol. 14, no. 1, pp. 119–126, mar 2006. [Online]. Available: http://www.mitpressjournals.org/doi/10.1162/evco.2006.14.1.119