Information Sciences 605 (2022) 15–42

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

SMGO-Δ: Balancing caution and reward in global optimization with black-box constraints

Lorenzo Sabug Jr. [a,1,*], Fredy Ruiz [a], Lorenzo Fagiano [a,2]

[a] Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, Piazza Leonardo da Vinci 32, 20133 Milano, Italy

ARTICLE INFO

Article history:
Received 9 November 2021
Received in revised form 29 March 2022
Accepted 3 May 2022
Available online 7 May 2022

Keywords:
Black-box optimization
Derivative-free optimization
Black-box constraints
Set membership

ABSTRACT

In numerous applications across all science and engineering areas, there are optimization problems where both the objective function and the constraints have no closed-form expression or are too complex to be managed analytically, so that they can only be evaluated through experiments. To address such issues, we design a global optimization technique for problems with black-box objective and constraints. Assuming Lipschitz continuity of the cost and constraint functions, a Set Membership framework is adopted to build a surrogate model of the optimization program, that is used for exploitation and exploration routines. The resulting algorithm, named Set Membership Global Optimization with black-box constraints (SMGO-Δ), features one tunable risk parameter, which the user can intuitively adjust to trade-off safety, exploitation, and exploration. The theoretical properties of the algorithm are derived, and the optimization performance is compared with representative techniques from the literature in several benchmarks. An extension to uncertain cost/constraint function outcomes is presented, too, as well as computational aspects. Lastly, the approach is tested and compared with constrained Bayesian optimization in a case study pertaining to model predictive control tuning for a servomechanism with disturbances and plant uncertainties, addressing practically-motivated task-level constraints.

1. Introduction

Black-box optimization problems arise in many practical scenarios, such as engineering design, management of complex processes, and control system tuning. A common aspect to such problems is that the objective function involves many interacting factors that are difficult to model mathematically in an accurate way. In these contexts, the objective function is evaluated using (possibly time-consuming) experiments. Another relevant case is when accurate mathematical models do exist, but they can not be easily treated analytically because of their high complexity: in these situations, one can usually rely only on simulations. Typically, these problems feature several local minima and they may also be affected by uncertainty/disturbances, so that the value of the cost may change within certain intervals for the same value of the decision variables. These

* Corresponding author.
E-mail addresses: lorenzojr.sabug@polimi.it (L. Sabug Jr.), fredy.ruiz@polimi.it (F. Ruiz), lorenzo.fagiano@polimi.it (L. Fagiano).

[1] 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.

[2] 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".

https://doi.org/10.1016/j.ins.2022.05.017
0020-0255/

aspects make the use of established first- or second-order nonlinear programming approaches impractical. Rather, global black-box optimization techniques aim to solve these problems using the results of the experiments/simulations to iteratively choose the value of the decision variable to be tested next, attempting to simultaneously learn the objective function and optimize it. Almost all such techniques try to balance exploitation and exploration, in order to ensure fast convergence and coverage of the search space, respectively. Examples of algorithms for black-box optimization in the unconstrained case are [1–9].

Many black-box optimization problems involve also the satisfaction of one or more constraints, deriving for example from the fulfillment of norms and standards, the achievement of a minimum performance threshold, or state constraints. In the literature, most black-box optimization approaches, including the references above, do consider constraint functions that are assumed to be known *a priori* and either analytically or numerically tractable, such as upper and lower bounds on each decision variables, convex sets, or nonlinear but known and differentiable constraint functions. However, there are problems where the constraint values are not known *a priori* or too complicated and uncertain as well, i.e., they are also black box, leading to uncertain and possibly disconnected feasible sets. This paper focuses on this problem class. In particular, we consider a framework where sensible values of the objective and constraint functions are returned also when the sampled point is outside the feasible set, i.e., the black-box constraints can be violated during the optimization process. This is in contrast to non-relaxable problems, in which a violated constraint means a crashed simulation or a failed experiment, and the returned objective and/or constraint function values are undefined or not meaningful. Using the taxonomy of black-box constraints introduced in [10], we deal with `QRSK` (Quantifiable, Relaxable, Simulation, Known) problems.

1.1. Previous work

The interest on considering black-box constraints in black-box optimization is increasing, mainly in synchronous contexts, where the objective and all constraints are jointly evaluated at each sampling point. Efforts have also been made for different problem setups such as in multi-objective contexts [11–13], and in asynchronous evaluations (where the objective and/or constraints can be independently sampled at different test points) [13–15]. However, such contexts are outside the scope of the present work, as we focus on scalar-objective, synchronous constrained optimization.

Swarm- or population-based approaches were formulated in previous works to tackle the problem we treat. For example, [16] proposes the use of a penalty function, recasting the constrained problem to an unconstrained one, allowing the use of a hybrid between gravitational search and genetic algorithms. An approach proposed by [17] recasts the problem into a bi-objective one with the problem objective and overall constraint violation (defined as an average of the individual constraint violations). These population-based algorithms, however, use the sum (or a linear combination) of constraint violations as measure for infeasibility. This may lead to high sensitivity w.r.t. the size of the constraint function image sets, i.e. a constraint can dominate others because it has a wider range of values.

Another popular class of algorithms is based on the Mesh Adaptive Direct Search (MADS) [18–20], used in applications as reported in [21,22]. In general, the algorithm is composed of search (exploration) and poll (exploitation) phases, selecting evaluation points from a mesh emanating from the current best point, usually along the coordinate directions. Such a mesh becomes more refined as the iterations increase, until the evaluation budget is exhausted. It features flexibility in terms of the selection criteria for generating evaluation points, especially for the search phase. However, the mesh generation for the search phase is exponential w.r.t. dimensionality. Furthermore, the candidate points selection from a mesh limits the search directions that can be taken for evaluation.

In the last decade, kriging-based methods [23–26] and Bayesian optimization (BO) [27,28] grew in popularity. These techniques have a common base: fitting Gaussian Process (GP) priors to existing objective and constraint data, and selecting a point that maximizes an acquisition function over the fitted prior. Earlier works addressed the modification of acquisition function formulations considering the objective and constraints/feasibility [14,24,25,29–31]. Some works like [24,30] need feasible point/s in the initial data, but [14] addressed this issue by focusing on feasible region search when no feasible samples exist yet. [32] proposed a similar approach, devoting the initial iterations to learn the feasible regions, while [31] proposed alternating between exploitation (inside the feasible region) and exploration (discovering the constraint boundaries). Other works [33,34] leveraged in their sampling point selection the expected information gain w.r.t. existing sampled constrained minimal value. In later works, lookahead-based formulations [35,36] were investigated as well, arguing that one-step (or myopic/greedy) algorithms only consider immediate improvement of solution quality, and tend to ignore long-term information gains. A major limitation of such kriging- and Bayesian-based methods lies in the computationally-intensive GP fitting over the existing data, which only allows their use in low-dimensional problems (most previous literature demonstrated its use up to 5–6 dimensions only). Furthermore, optimizing the acquisition function over these "cheap" surrogate surfaces is another problem. Most techniques delegate the surrogate optimization to an external solver, which may affect reproducibility of results. Furthermore, while they show empirically good performance, convergence properties are not investigated for the above-discussed methods.

A recently-developed class of constrained algorithms, as in [37–39], utilizes a Delaunay triangulation to quantify the uncertainty throughout the search space, given the existing sampled points. These methods feature flexibility in using any smooth interpolation for the surrogate modeling of the objective and constraints. In addition, their convergence is proven assuming twice-differentiable objective and constraint functions. However, the heavy computational burden of calculating/updating a Delaunay triangulation practically limits their use to low-dimension problems.

In summary, while existing methods of constrained black-box optimization have shown good performance, computational burden issues are still pertinent. This limits the practical use of black-box optimization, especially in higher-dimensional problems and in embedded hardware with limited computational facilities. The reproducibility of results – a mostly-ignored aspect in previous literature – is also highly relevant in industrial contexts, in which explainable and traceable history of experiments is desirable. A organized summary comparing the previous work is shown in Table 1.

1.2. Contributions of this work

The present paper proposes a new approach, named Set Membership Global Optimization with black-box constraints (SMGO-Δ), which is computationally efficient, provably convergent, reproducible, and competitive in iteration-based performance. Deepening the research line of our recent work [7,8] which only considered unconstrained cases, we use the Set Membership framework [40] to build estimated models of both the cost and the constraints. These approximations are then used to predict the fulfillment of the black-box constraints in the unsampled areas. A preliminary version of SMGO-Δ has been presented in [41], and the results and discussions have been significantly expanded for this paper. The source code for the presented method is available in https://github.com/lorenzosabugjr/smgo-delta. Our contributions in this paper are as follows:

- We present a computationally efficient estimation model for the objective and constraints, given existing data. The Set Membership (SM)-based model is geometrically based on hypercones, which quantify the uncertainty in a more tractable way than GP-based fitting, and Delaunay partitioning-based methods.
- The algorithm introduces exploitation and exploration strategies and sampling point selection based on methodically-generated candidate points, which eliminates the need for external techniques such as swarm optimizers to optimize over the SM response surface. The candidate points generation method allows us to consider a generalized convex search space, in contrast to that of [7,8] which only applies to convex polytopes. The points generation proposed in SMGO-Δ offers richer variety of search directions than MADS [18–20]. Furthermore, it allows the algorithm to provide reproducible results, i.e. the same starting sample/s will lead to the same best result, assuming no noise.
- We introduce a custom-tunable parameter to change the exploration behavior, ranging from "cautious" (staying in the currently discovered feasible region) to "risky" (allowing discovery of disjoint feasible regions, which might contain better points or even the global minimizer). Furthermore, we do not require an initial feasible point, and we automatically prioritize regions where more constraints are estimated as fulfilled, encouraging the search for the feasible region.
- The theoretical convergence of the proposed algorithm – an aspect that is often not rigorously considered in the literature – is investigated and proven in this paper. The convergence guarantees we give are only subject to Lipschitz continuity assumption for the objective and constraints, milder than in [37–39].
- We treat practical implementation concerns w.r.t. bounded noise and search directions, and offer methods to deal with such. Furthermore, the computational complexity of the method is analyzed, and with an iterative implementation, we can improve the complexity of a single iteration to $\mathcal{O}(Dn + n^2)$, where D is the search space dimensionality, and n is the number of existing samples.
- The presented SMGO-Δ is compared with other optimizers in a set of benchmark functions, and is shown to be competitive w.r.t. the state of the art. Furthermore, we compare it side-by-side with constrained Bayesian optimization (CBO) in an engineering problem of black-box tuning of a model predictive controller (MPC) for a servomotor. We show via statistical tests that our proposed method is competitive with CBO in iteration-wise optimization performance, but is much faster in terms of computational time.

Table 1
Comparison of representative methods for global optimization with black-box constraints.

Type	Advantages	Disadvantages
Swarm-based [16,17]	Simple implementation and concept of penalty function; computationally efficient; repeatable	Sensitive to value ranges of respective constraints; convergence not investigated
MADS [18–20]	Simple design; high flexibility in global search criteria	Lack of search directions due to meshing, complexity of mesh generation is exponential w.r.t. dimensionality
Kriging- [23–26] or BO-based [14,27–30]	Iteration-efficient; wide implementation	Computationally demanding; practically limited to low dimensions; convergence not investigated; not completely repeatable
Delaunay-based [37–39]	Iteration-efficient; flexible to different regression techniques; provably convergent	Computationally heavy; practically limited to low dimensions
SMGO-Δ (this work)	Iteration- and computationally efficient; simple implementation; provably convergent; repeatable; user-tunable exploration behavior	Not memory-efficient; less effective with very small feasible regions

This paper is organized as follows. Section 2 states the assumptions and the problem we aim to address. The proposed method is introduced in Section 3, followed by the theoretical properties in Section 4. Section 5 presents the extension to the case of uncertain/noisy function evaluations and a discussion on implementation aspects. The performance and sensitivity analysis with an analytic function are presented in Section 6. Benchmarks and comparison with state-of-the-art techniques are given in Section 7. The results of the MPC tuning case study are discussed in Section 8, and we draw the conclusions in Section 9.

2. Problem statement

We consider the minimization of a cost function $f(\boldsymbol{x}), f(\boldsymbol{x}) : \mathscr{X} \to \mathbb{R}$, where $\mathscr{X} \subset \mathbb{R}^D$ is a compact and convex search set (if only black-box constraints are present, $\mathscr{X}$ can be taken for example as a large-enough bounding hypercube containing all values of $\boldsymbol{x}$ that are meaningful according to the application at hand). This minimization is subject to the satisfaction of constraints $g_s, s = 1, \ldots, S$: a constraint g_s is satisfied at point $\boldsymbol{x}$ when $g_s(\boldsymbol{x}) \geqslant 0$.

Neither f nor any g_s are assumed to be known. The only *a priori* assumption about f and all g_s is given as follows:

Assumption 1. f and $g_s, s = 1, \ldots, S$ are Lipschitz continuous functions over $\mathscr{X}$ with unknown Lipschitz constants $\gamma, \rho_1, \ldots, \rho_S$:

$$\begin{aligned} f &\in \mathscr{F}(\gamma) \\ g_1 &\in \mathscr{F}(\rho_1) \\ &\vdots \\ g_S &\in \mathscr{F}(\rho_S) \end{aligned}$$

where

$$\mathscr{F}(\eta) \doteq \{h : |h(\boldsymbol{x}_1) - h(\boldsymbol{x}_2)| \leqslant \eta \|\boldsymbol{x}_1 - \boldsymbol{x}_2\|, \forall \boldsymbol{x}_1, \boldsymbol{x}_2 \in \mathscr{X}\}. \tag{1}$$

In this paper, $\|\cdot\|$ is the 2-norm (or the Euclidean norm). We also assume to be able to evaluate these functions at a given point $\boldsymbol{x}^{(n)}$, where $n \in \mathbb{N}$ is a counter of sampled data points. We denote the resulting values as $z^{(n)} = f(\boldsymbol{x}^{(n)})$ and $c_s^{(n)} = g_s(\boldsymbol{x}^{(n)})$. This is reasonable in many cases of practical interest, when physical processes have finite sensitivity (Assumption 1), the process is repeatable, and accurate sensors are used. In Section 5, we consider an extension to the case of unknown but bounded additive uncertainty.

For compactness of notation, we also introduce the vector of sampled constraint values as $\boldsymbol{c}^{(n)} = \left[c_1^{(n)}, \ldots, c_S^{(n)}\right]^\top$, where $\top$ denotes the matrix transpose operation. Finally, we assume a non-empty feasible set. Let

$$\mathscr{G}_s = \{\boldsymbol{x} \in \mathscr{X} : g_s(\boldsymbol{x}) \geqslant 0\}$$

be the set of points satisfying the s-th constraint.

Assumption 2. Denoting $\mathscr{G} \triangleq \mathscr{X} \cap \left\{\bigcap_{s=1}^{S} \mathscr{G}_s\right\}$, we have

$$\mathscr{G} \neq \varnothing.$$

Due to Assumptions 1 and 2, there exists at least one global minimizer $\boldsymbol{x}^*$ such that

$$\boldsymbol{x}^* \in \{\boldsymbol{x} \in \mathscr{G} | \forall \boldsymbol{x}\prime \in \mathscr{G}, f(\boldsymbol{x}\prime) \geqslant f(\boldsymbol{x})\}.$$

with the corresponding minimum $z^* = f(\boldsymbol{x}^*)$. We denote the data collected by our optimization algorithm as

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

The tuple describing the best (feasible) sample $\left(\boldsymbol{x}^{*\langle n \rangle}, z^{*\langle n \rangle}, \boldsymbol{c}^{*\langle n \rangle}\right)$ is

$$\begin{aligned} \left(\boldsymbol{x}^{*\langle n \rangle}, z^{*\langle n \rangle}, \boldsymbol{c}^{*\langle n \rangle}\right) \quad = \quad \arg \quad & \min_{\left(\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}\right) \in \boldsymbol{X}^{\langle n \rangle}} z^{(i)} \\ & \text{s.t. } \boldsymbol{c}^{(i)} \geqslant 0. \end{aligned} \tag{3}$$

A lexicographic criterion is used to sort out possible multiple feasible points with the same (best) cost. In the remainder, for the sake of notational simplicity, we refer to the best point as $\boldsymbol{x}^{*\langle n \rangle}$. We make no assumptions about the feasibility of the starting point and the existence of the best point $\boldsymbol{x}^{*\langle n \rangle}$ (as defined in (3)) at each iteration. Hence, $\boldsymbol{x}^{*\langle n \rangle}$ may not exist for the first iterations of the procedure, until for some n the test point is feasible, after which it will exist for all succeeding iterations. The theoretical properties of SMGO-Δ, which we introduce later on, guarantee that indeed a feasible point will be sampled at finite iterations, under Assumption 2.

Now we are ready to state the problem addressed in this paper.

Problem 1. Design an algorithm that generates a sequence of points $\{\boldsymbol{x}^{(1)}, \boldsymbol{x}^{(2)}, \ldots\}, \boldsymbol{x}^{(i)} \in \mathscr{X}$, such that

$$\forall \varepsilon > 0, \exists n_\varepsilon < \infty : z^{*\langle n_\varepsilon \rangle} \leqslant z^* + \varepsilon,$$
$$c_s^{*\langle n_\varepsilon \rangle} \geqslant 0, s = 1, \ldots, S.$$

3. Set Membership Global Optimization with black-box constraints (SMGO-): Algorithm

As usual in the literature, the search for $\boldsymbol{x}^*$ is carried out by a sequential sampling procedure, wherein the next test point is decided on the basis of the existing data. The inclusion of black-box constraints implies significant changes in the algorithm design w.r.t. the unconstrained case [8], as discussed in the following subsections.

3.1. Update of the data-set, and of the Lipschitz constants estimates

At each iteration n, a point $\boldsymbol{x}^n \in \mathscr{X}$ is tested by the algorithm, chosen with a strategy described later on. The corresponding sampled tuple $\left(\boldsymbol{x}^n, z^{(n)}, \boldsymbol{c}^{(n)}\right) \in \mathbb{R}^{D+1+S}$ is added to the data-set $X^{\langle n \rangle}$:

$$\boldsymbol{X}^{\langle n \rangle} = \boldsymbol{X}^{\langle n-1 \rangle} \cup \left(\boldsymbol{x}^n, z^{(n)}, \boldsymbol{c}^{(n)}\right).$$

From the updated data-set $\boldsymbol{X}^{\langle n \rangle}$, the algorithm updates the Lipschitz constants' estimates $\tilde{\gamma}^{\langle n \rangle}$ and $\tilde{\rho}_s^{\langle n \rangle}, s = 1, \ldots, S$ (see Assumption 1) as (see also [40]):

$$\tilde{\gamma}^{\langle n \rangle} = \max_{\left(\boldsymbol{x}^i, z^{(i)}, \boldsymbol{c}^i\right), \left(\boldsymbol{x}^j, z^{(j)}, \boldsymbol{c}^j\right) \in \boldsymbol{X}^{\langle n \rangle}} \left(\frac{|z^{(i)} - z^{(j)}|}{\|\boldsymbol{x}^i - \boldsymbol{x}^j\|}, \underline{\gamma} \right), \tag{4}$$

$$\tilde{\rho}_s^{\langle n \rangle} = \max_{\left(\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}\right), \left(\boldsymbol{x}^{(j)}, z^{(j)}, \boldsymbol{c}^{(j)}\right) \in \boldsymbol{X}^{\langle n \rangle}} \left(\frac{|c_s^{(i)} - c_s^{(j)}|}{\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|}, \underline{\rho}_s \right), \tag{5}$$

where $\underline{\gamma} > 0$, $\underline{\rho}_s > 0$ are small but finite initial estimates for γ, ρ_s, $s = 1, \ldots, S$, respectively.

The above-given estimated Lipschitz constants are, by construction, unfalsified by the available data. At $n = 1$, one can set these estimates to $\underline{\gamma}, \underline{\rho}_s$, and select the test point $\boldsymbol{x}^{(1)}$ with a strategy of choice (for example a random starting point, or, as discussed in our simulation example, a sensible point for the application at hand).

We can now build the following upper- and lower-bound functions, $\overline{f}^{\langle n \rangle}(\boldsymbol{x})$ and $\underline{f}^{\langle n \rangle}(\boldsymbol{x})$, resorting to a Set Membership approach [40]:

$$\overline{f}^{\langle n \rangle}(\boldsymbol{x}) \triangleq \min_{k=1,\ldots,n} \left(z^{(k)} + \tilde{\gamma}^{\langle n \rangle} \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \right), \tag{6}$$

$$\underline{f}^{\langle n \rangle}(\boldsymbol{x}) \triangleq \max_{k=1,\ldots,n} \left(z^{(k)} - \tilde{\gamma}^{\langle n \rangle} \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \right). \tag{7}$$

These functions represent the tightest bounds of the cost function value in the unsampled regions, given the sampled points and the Lipschitz continuity assumption. Furthermore, we build the central approximation of the objective function,

$$\tilde{f}^{\langle n \rangle}(\boldsymbol{x}) = \frac{1}{2}\left(\overline{f}^{\langle n \rangle}(\boldsymbol{x}) + \underline{f}^{\langle n \rangle}(\boldsymbol{x}) \right)$$

and the uncertainty measure

$$\lambda^{\langle n \rangle}(\boldsymbol{x}) = \overline{f}^{\langle n \rangle}(\boldsymbol{x}) - \underline{f}^{\langle n \rangle}(\boldsymbol{x}).$$

A visual interpretation of the SM-based bounds and function approximation is shown in Fig. 1. The same is performed to calculate the quantities $\overline{g}_s^{\langle n \rangle}(\boldsymbol{x}), \underline{g}_s^{\langle n \rangle}(\boldsymbol{x}), \tilde{g}_s^{\langle n \rangle}(\boldsymbol{x})$, and $\pi_s^{\langle n \rangle}(\boldsymbol{x})$ for each constraint function g_s:

$$\overline{g}_s^{\langle n \rangle}(\boldsymbol{x}) \triangleq \min_{k=1,\ldots,n} \left(c_s^{(k)} + \tilde{\rho}_s^{\langle n \rangle} \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \right) \tag{8a}$$

$$\underline{g}_s^{\langle n \rangle}(\boldsymbol{x}) \triangleq \max_{k=1,\ldots,n} \left(c_s^{(k)} - \tilde{\rho}_s^{\langle n \rangle} \|\boldsymbol{x} - \boldsymbol{x}^{(k)}\| \right) \tag{8b}$$

$$\tilde{g}_s^{\langle n \rangle}(\boldsymbol{x}) = \frac{1}{2}\left(\overline{g}_s^{\langle n \rangle}(\boldsymbol{x}) + \underline{g}_s^{\langle n \rangle}(\boldsymbol{x}) \right) \tag{8c}$$

$$\pi_s^{\langle n \rangle}(\boldsymbol{x}) = \overline{g}_s^{\langle n \rangle}(\boldsymbol{x}) - \underline{g}_s^{\langle n \rangle}(\boldsymbol{x}). \tag{8d}$$

After updating the current best point and the estimates of Lipschitz constant, upper and lower bounds, functions values and uncertainty, the proposed optimization algorithm attempts first an exploitation strategy, possibly followed (if not satisfactory enough) by an exploration one. These two routines are described next.

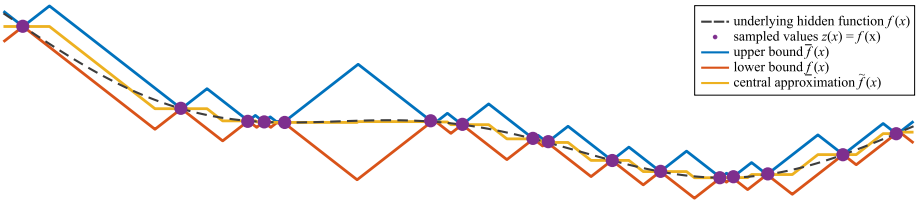

Fig. 1. SM-based upper- and lower bounds, and function approximation.

3.2. Exploitation with constraints

This subroutine attempts to improve on the current best value by searching $\mathscr{X}$ for a better candidate point according to the lower bound of the cost function, while satisfying the estimated constraints. Let us denote with $\boldsymbol{E}^{\langle n\rangle} \in \mathscr{X}$ a finite set of candidate points, selected according to a gridding strategy described in Section 3.4, and with $\mathscr{T}^{\langle n\rangle}$ a trust region around $\boldsymbol{x}^{*\langle n\rangle}$, discussed in Section 3.5. Let us further introduce the user-chosen scalar $\Delta \in [0, 1]$. Then, the exploitation routine solves the following problem:

$$\boldsymbol{x}_\theta^{\langle n\rangle} = \arg \min_{\boldsymbol{x} \in \boldsymbol{E}^{\langle n\rangle} \cap \mathscr{T}^{\langle n\rangle}} \xi_\theta(\boldsymbol{x}) \tag{9a}$$

$$\text{s.t. } \Delta \tilde{g}_s^{\langle n\rangle}(\boldsymbol{x}) + (1-\Delta)\underline{g}_s^{\langle n\rangle}(\boldsymbol{x}) \geqslant 0, \quad s = 1, \ldots, S \tag{9b}$$

where the exploitation cost is defined as

$$\xi_\theta(\boldsymbol{x}) \triangleq \tilde{f}^{\langle n\rangle}(\boldsymbol{x}) - \beta \hat{\lambda}^{\langle n\rangle}(\boldsymbol{x}), \tag{10}$$

and β is a user-defined weighting parameter. The objective function in (9a) is a trade-off between minimizing the central approximation of the cost function f ($\beta = 0$), and maximizing the associated uncertainty in order to gain more information with the next sample ($\beta > 0$). As default value, we set $\beta = 0.1$ as in [41].

We confine the optimization (9a) inside a trust region $\mathscr{T}^{\langle n\rangle}$ to search in the vicinity of the existing best point $\boldsymbol{x}^{*\langle n\rangle}$, evaluating whether there is a neighboring point that improves on the current best. Furthermore, our design of exploitation cost $\xi_\theta(\boldsymbol{x})$ aims to choose a point that improves as much as possible from $\boldsymbol{x}^{*\langle n\rangle}$, by using the lower bounds. Resolving the terms in (10), it is actually a weighted sum of $\bar{f}^{\langle n\rangle}(\boldsymbol{x})$ and $\underline{f}^{\langle n\rangle}(\boldsymbol{x})$ which results in favoring points that are close to $\boldsymbol{x}^{*\langle n\rangle}$, aiming for a local optimization.

The hyperparameter Δ in (9b) is referred to as the *risk factor*: it affects the cautiousness in choosing the exploitation point. $\Delta = 0$ provides the most cautious choice of a feasible candidate point, because it uses the constraints' lower bounds $\underline{g}_s^{\langle n\rangle}(\boldsymbol{x})$ to estimate the feasible set, thereby shrinking it considerably. On the other hand, $\Delta = 1$ is associated with a riskier exploitation behavior, using the central approximation $\tilde{g}_s^{\langle n\rangle}(\boldsymbol{x})$ and correspondingly expanding the estimated feasible set.

If (9a) is estimated to be feasible, the exploitation routine then carries out a test on the expected improvement obtained by the resulting point $\boldsymbol{x}_\theta^{\langle n\rangle}$, w.r.t. the current best value, as done also in [8]:

$$\underline{f}^{\langle n\rangle}\left(\boldsymbol{x}_\theta^{\langle n\rangle}\right) \leqslant z^{*\langle n\rangle} - \eta, \tag{11}$$

where $\eta = \alpha \tilde{\gamma}^{\langle n\rangle}$ is the expected improvement threshold. If this test is passed, SMGO-Δ will choose to sample the selected point, i.e. $\boldsymbol{x}^{\langle n+1\rangle} = \boldsymbol{x}_\theta^{\langle n\rangle}$. On the other hand, if (9a) is estimated infeasible or condition (11) is not met, the optimization algorithm proceeds to exploration. Through the expected improvement test, SMGO-Δ avoids being stuck in a local minimum, as will be proven in Section 4.

Remark 1. In this paper, for simplicity we loosely use the term "feasibility" and "constraint satisfaction" depending on the context. When we describe a sample $\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n\rangle}$ as being feasible or having satisfied/fulfilled a constraint g_s, we mean that the constraints are actually satisfied as evaluated in the corresponding simulation/experiment, i.e., $c_s^{(i)} \geqslant 0$. However, we consider that an unsampled point $\boldsymbol{x} \in \mathscr{X} \setminus \boldsymbol{X}^{\langle n\rangle}$ is feasible or satisfies constraint g_s if it is predicted to fulfill the constraint/s based on its central estimate (8c), i.e., $\tilde{g}_s(\boldsymbol{x}) \geqslant 0$.

3.3. Exploration by uncertainty and estimated constraint violations

The exploration routine probes the areas of the search space where cost function uncertainty is largest, while at the same time penalizing possible constraint violations. The exploration point $\boldsymbol{x}_\psi^{\langle n\rangle}$ is chosen as:

$$\boldsymbol{x}_{\psi}^{\langle n\rangle} = \arg\max_{\boldsymbol{x}\in\boldsymbol{E}^{\langle n\rangle}} \xi_{\psi}^{\langle n\rangle}(\boldsymbol{x}) \tag{12}$$

with the *exploration merit function* defined as:

$$\xi_{\psi}^{\langle n\rangle}(\boldsymbol{x}) \triangleq h^{\langle n\rangle}(\boldsymbol{x}) + k\big(\tau^{\langle n\rangle}(\boldsymbol{x})\big), \tag{13}$$

where $\tau^{\langle n\rangle}(\boldsymbol{x})$ describes the age of the candidate point, i.e. the iterations elapsed since its creation. Furthermore, $k(\cdot)$ must be chosen as a class-$\mathscr{K}_{\infty}$ function, i.e. such that it is continuous, strictly increasing, $\lim_{\tau\to\infty} k(\tau) = +\infty$, and $k(0) = 0$. The inclusion of $k(\cdot)$ encourages the exploration of the most remote areas in $\mathscr{X}$, and in turn, guarantees the convergence of SMGO-Δ, as proven in Section 4. Finally, $h^{\langle n\rangle}(\cdot)$ is a user-chosen merit function, whose sole requirement is to be bounded, i.e., that $\forall \boldsymbol{x} \in \mathscr{X}, n \in [1, +\infty), |h^{\langle n\rangle}(\boldsymbol{x})| < \infty$. In principle, this function shall be chosen to trade-off constraint satisfaction and exploration of points where uncertainty is largest. This is in contrast with our work in the unconstrained case [7,8] where we only search for points with highest uncertainty. In this work, we propose the following:

$$h^{\langle n\rangle}(\boldsymbol{x}) \triangleq d(\boldsymbol{x})\big((1-\Delta)w_{\lambda}(\boldsymbol{x}) + \Delta w_{\pi}(\boldsymbol{x})w_{g}(\boldsymbol{x})\big), \tag{14}$$

where:

$$d(\boldsymbol{x}) = \min_{\left(\boldsymbol{x}^{(i)},z^{(i)},\boldsymbol{c}^{(i)}\right)\in\boldsymbol{X}^{\langle n\rangle}} \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\|, \tag{15}$$

$$w_{\lambda}(\boldsymbol{x}) = \begin{cases} \lambda^{\langle n\rangle}(\boldsymbol{x}) & \text{if } \Delta\tilde{\underline{g}}_{s}^{\langle n\rangle}(\boldsymbol{x}) + (1-\Delta)\underline{g}_{s}^{\langle n\rangle}(\boldsymbol{x}) \geqslant 0, s = 1,\ldots,S \\ 0 & \text{otherwise,} \end{cases} \tag{16}$$

$$w_{\pi}(\boldsymbol{x}) = \sum_{s=1}^{S} \frac{\pi_{s}^{\langle n\rangle}(\boldsymbol{x})}{\tilde{\rho}_{s}^{\langle n\rangle}}, \tag{17}$$

$$w_{g}(\boldsymbol{x}) = 2^{-S+\sum \mathbb{1}_{s}(\boldsymbol{x})} \tag{18}$$

$$\mathbb{1}_{s}(\boldsymbol{x}) = \begin{cases} 1 & \text{if } \tilde{\underline{g}}_{s}^{\langle n\rangle}(\boldsymbol{x}) \geqslant 0 \\ 0 & \text{otherwise.} \end{cases} \tag{19}$$

The merit function prioritizes more remote candidate points by using $d(\boldsymbol{x})$ (15) as a multiplier to the other terms in $h^{\langle n\rangle}(\boldsymbol{x})$. The term $w_{\lambda}(\boldsymbol{x})$ in (14) is included to encourage the exploration of points with more uncertainty w.r.t. the cost function f, using the same condition as (9b), hence exhibiting cautious exploration depending on Δ. This is motivated by our need to learn the underlying cost f since sampling will shrink the uncertainty $\lambda^{\langle n\rangle}$ at that point to zero, while correspondingly decreasing the uncertainty in its vicinity. However, we only consider points that we estimate to be feasible, to avoid sampling in unfeasible regions when we already discovered feasible one/s.

The second term in (14) aims to discover the feasible set, favoring points with higher uncertainty $\pi^{\langle n\rangle}s$ w.r.t. the constraint functions g_s. Each estimate $\pi_{s}^{\langle n\rangle}$ is normalized by its Lipschitz constant $\tilde{\rho}_{s}^{\langle n\rangle}$ (see (17)) to be less sensitive with respect to the range of absolute values of each constraint.

We define (18)-(19) to give priority to points with more (estimated) satisfied constraints, practically doubling the merit with every additional constraint fulfilled at a point.

The risk factor $\Delta \in [0, 1]$, in a similar manner as in exploitation, affects the level of caution in exploring outside the estimated feasible region. $\Delta \to 0$ results in higher cautiousness, favoring sampling within the estimated feasible region (we mostly consider $w_{\lambda}(\boldsymbol{x})$ in our exploration).

The selected point $\boldsymbol{x}_{\psi}^{\langle n\rangle}$ is then directly assigned as the next sample point $\boldsymbol{x}^{\langle n+1\rangle}$.

Remark 2. The concept of *caution* considered here is different from *safe* optimization as discussed in [42]. Indeed, because the Lipschitz constants ρ_s of the constraints g_s are unknown, we cannot guarantee *a priori* the safety of a test point. On the other hand, under the additional assumptions of having a feasible starting point and of knowing upper bounds on the Lipschitz constants ρ_s, one can use $\Delta = 0$ to guarantee robustly hard constraint satisfaction.

Remark 3. We have introduced the age-based term in (13). While this guarantees convergence, a high slope for k can cause candidate points to be chosen because of their age, and not because of their potential information gain. Thus, k should be chosen with very small slope to mitigate this effect.

3.4. Generation of candidate points

In the selection of the exploitation point $\boldsymbol{x}_{0}^{\langle n\rangle}$ (9a) and the exploration one $\boldsymbol{x}_{\psi}^{\langle n\rangle}$ (12), we consider a set of candidates $\boldsymbol{E}^{\langle n\rangle}$, which is systematically generated based on the existing data. Such approximation eliminates the need for an external

optimization algorithm to calculate (9a) and (12). Furthermore, the proposed algorithm is repeatable, i.e. given the same starting point $\boldsymbol{x}^{\langle 1 \rangle}$ and search set $\mathscr{X}$, the algorithm will produce the same sampling points sequence and final result.

In [7,8], we proposed a candidate point generation built upon the midpoints among all existing samples, including the vertices of the search set, assumed to be polytopic in those previous works. However, the complexity of such candidate points generation is exponential w.r.t. dimension D, in the rather common case of hyperrectangular $\mathscr{X}$. Hence, we propose another candidate points generation strategy with improved complexity w.r.t. D.

Given an incoming sample $\boldsymbol{x}^n$, we cumulatively add candidate points to $\boldsymbol{E}^{\langle n \rangle}$ according to the following criteria:

1. along the coordinate directions stemming from $\boldsymbol{x}^n$,
2. by gridding between the sampled point and all the other ones.

To generate 1), using the coordinate directions $\pm\hat{\boldsymbol{a}}_d, d = 1, \ldots, D$, we define

$$\begin{aligned} \boldsymbol{b}_{+d}^{(n)} &= \max_{b \in [0,\infty)} b \\ \text{s.t.} \quad & \boldsymbol{x}^n + b\hat{\boldsymbol{a}}_d \in \mathscr{X}, \end{aligned} \tag{20}$$

$$\begin{aligned} \boldsymbol{b}_{-d}^{(n)} &= \max_{b \in [0,\infty)} b \\ \text{s.t.} \quad & \boldsymbol{x}^n - b\hat{\boldsymbol{a}}_d \in \mathscr{X}. \end{aligned} \tag{21}$$

These are the lengths of the longest segments inside the search set $\mathscr{X}$ departing from $\boldsymbol{x}^n$ in the coordinate directions. Then, the sets of candidate points generated by $\boldsymbol{x}^n$ along the coordinate directions are

$$\boldsymbol{Y}_{+d}^{(n)} = \left\{ \boldsymbol{x}^n + \frac{k}{B} \boldsymbol{b}_{+d}^{(n)} \hat{\boldsymbol{a}}_d, \ k \in \{1, \ldots, B-1\} \right\} \tag{22}$$

$$\boldsymbol{Y}_{-d}^{(n)} = \left\{ \boldsymbol{x}^n - \frac{k}{B} \boldsymbol{b}_{-d}^{(n)} \hat{\boldsymbol{a}}_d, \ k \in \{1, \ldots, B-1\} \right\} \tag{23}$$

where $1/B$ is the relative grid granularity.

For criterion 2), we define the candidate points along the segments from $\boldsymbol{x}^n$ pointing to each other sample $\boldsymbol{x}^{(j)}, j = 1, \ldots, n-1$ as

$$\boldsymbol{Y}_{\circ}^{(n)} = \left\{ \boldsymbol{x}^n + \frac{k}{B} \boldsymbol{x}^{(j)}, \ k \in \{1, \ldots, B-1\}, \ j \in \{1, \ldots, n-1\} \right\}.$$

Finally, the set of candidate points generated by $\boldsymbol{x}^n$ is

$$\boldsymbol{Y}^{(n)} = \left(\bigcup_{d=1}^{D} \left\{ \boldsymbol{Y}_{+d}^{(n)}, \boldsymbol{Y}_{-d}^{(n)} \right\} \right) \bigcup \boldsymbol{Y}_{\circ}^{(n)}. \tag{24}$$

Lastly, the aggregate collection of candidate points at iteration n is now

$$\boldsymbol{E}^{\langle n \rangle} = \boldsymbol{E}^{\langle n-1 \rangle} \cup \boldsymbol{Y}^{(n)}.$$

This results in a cumulative total of $n(B-1)\left(2D + \frac{n-1}{2}\right)$ candidate points, which is polynomial w.r.t. n, but linear with D, thus computationally advantageous with higher search space dimensions. Furthermore, unlike our proposal in [8], we do not require anymore a polytopic search set $\mathscr{X}$, but only a convex and compact one.

3.5. Exploitation trust region $\mathscr{T}^{\langle n \rangle}$

The set $\mathscr{T}^{\langle n \rangle}$ in (9a) limits the choice of the exploitation candidate point to a trust region around $\boldsymbol{x}^{*\langle n \rangle}$.

A ball $\mathscr{T}^{\langle n \rangle}$, centered at $\boldsymbol{x}^{*\langle n \rangle}$ and with a radius $\upsilon^{\langle n \rangle} < 1$, is declared when a feasible best point $\boldsymbol{x}^{*\langle n \rangle}$ is found,

$$\mathscr{T}^{\langle n \rangle} = \left\{ \boldsymbol{x} \in \mathscr{X} : \left\| \boldsymbol{x} - \boldsymbol{x}^{*\langle n \rangle} \right\|_{\infty} \leqslant \upsilon^{\langle n \rangle} \right\} \tag{25}$$

In (25), any norm can be used, e.g. 1- or 2-norm, for example in order to adapt to the characteristics of the search set $\mathscr{X}$. The size of $\mathscr{T}^{\langle n+1 \rangle}$ is updated depending on the *a posteriori* value of the sample $z^{\langle n+1 \rangle}$, as follows:

$$\upsilon^{\langle n+1 \rangle} = \begin{cases} \min\left(\underline{\upsilon}, \kappa \upsilon^{\langle n \rangle}\right) & \text{if exploration was done in } n+1 \text{ or } z^{\langle n+1 \rangle} > z^{*\langle n \rangle} \\ \max\left(\underline{\upsilon}, \frac{1}{\kappa} \upsilon^{\langle n \rangle}\right) & \text{if exploitation was done in } n+1 \text{ and } z^{\langle n+1 \rangle} \leqslant z^{*\langle n \rangle} - \alpha \gamma^{\langle n \rangle}, \\ & \text{with } c^{\langle n+1 \rangle} s \geqslant 0, \quad s = 1, \ldots, S, \\ \upsilon^{\langle n \rangle} & \text{otherwise.} \end{cases} \tag{26}$$

where $\kappa < 1$ is the shrinking factor. In summary, we shrink the trust region when an exploitation fails, i.e. there is no exploitation improvement w.r.t. current best $z^{*\langle n \rangle}$, or if no exploitation point was found at iteration $n+1$ (exploration was

done instead). Conversely, we expand the trust region (up to $\overline{\upsilon}$) when the new sample from exploitation has *a posteriori* returned a feasible point and satisfied the expected improvement threshold.

The minimum size of $\mathscr{T}^{\langle n \rangle}$ is also limited, that is, if $\upsilon^{\langle n \rangle} \leqslant \underline{\upsilon}$, we reset the trust region size $\upsilon^{\langle n+1 \rangle} = \underline{\upsilon}$. In this paper we set the default trust region-related quantities as $\kappa = 0.5, \overline{\upsilon} = 0.1$, and $\underline{\upsilon} = \kappa^{10}\overline{\upsilon}$.

3.6. Algorithm summary

A pseudo-code of the proposed method is shown in Algorithm 1.

Algorithm 1: SMGO-Δ Algorithm

Input: Initial point $\boldsymbol{x}^{(1)}$, search space $\mathcal{X}$
Lipschitz constants estimates $\tilde{\gamma}^{\langle 1 \rangle} = \underline{\gamma}$, $\tilde{\rho}_s^{\langle 1 \rangle} = \underline{\rho}$
Maximum number of iterations N.
Parameters $\alpha, \beta, \psi, \Delta$

while *iteration n within the budget N* **do**
  // Objective, constraints evaluation and data update
  Evaluate the objective f and constraints g_s at $\boldsymbol{x}^{(n)}$, add the resulting sample $(\boldsymbol{x}^{(n)}, z^{(n)}, \boldsymbol{c}^{(n)})$ to the set $\boldsymbol{X}^{\langle n \rangle}$
  Update Lipschitz constants $\tilde{\gamma}^{\langle n \rangle}$, $\tilde{\rho}_1^{\langle n \rangle}, \ldots, \tilde{\rho}_S^{\langle n \rangle}$ according to (4)-(5), the current best sample $(\boldsymbol{x}^{*\langle n \rangle}, z^{*\langle n \rangle}, \boldsymbol{c}^{*\langle n \rangle})$ from $\boldsymbol{X}^{\langle n \rangle}$ (3), candidate points $\boldsymbol{E}^{\langle n \rangle}$, and the trust region $\mathcal{T}^{\langle n \rangle}$ (25)-(26)
  // Exploitation routine
  Solve (9a) to choose the candidate exploitation point $\boldsymbol{x}_\theta^{\langle n \rangle}$ in the estimated feasible region
  if $\boldsymbol{x}_\theta^{\langle n \rangle}$ *exists and expected improvement condition* (11) *is met* **then**
    Assign test point for next iteration $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_\theta^{\langle n \rangle}$
  else
    // Exploration routine
    Solve (12) to compute $\boldsymbol{x}_\psi^{\langle n \rangle}$
    Assign test point for next iteration $\boldsymbol{x}^{(n+1)} \leftarrow \boldsymbol{x}_\psi^{\langle n \rangle}$
  Go to next iteration $n \leftarrow n + 1$
Final optimal point and value: return the best sample $(\boldsymbol{x}^{*\langle N \rangle}, z^{*\langle N \rangle}, \boldsymbol{c}^{*\langle N \rangle})$ from the set $\boldsymbol{X}^{\langle N \rangle}$

4. Algorithm analysis

In this section, we analyze the properties of SMGO-Δ, and provide theoretical guarantees on its convergence to the best feasible point, by proving its dense points generation behavior.

In the remainder, given $\boldsymbol{x} \in \mathbb{R}^D$ and $r > 0$, we denote $\mathscr{B}(\boldsymbol{x}, r)$ as

$$\mathscr{B}(\boldsymbol{x}, r) \triangleq \{\boldsymbol{y} | \|\boldsymbol{y} - \boldsymbol{x}\| \leqslant r\}.$$

We now present the first lemma, showing that the exploitation will end in finite iterations, allowing for SMGO-Δ to escape local minima, and continue with exploration around the search space.

Lemma 1. *Exploitation will end after finite iterations.*

Proof. Exploitation ends, and the algorithm switches to exploration, when any of the following occurs:

1. No supposedly feasible point has been found so far: $\forall (\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}) \in \boldsymbol{X}^{\langle n \rangle}, \exists s \in [1, S] : c_s^{(i)} < 0$,
2. A feasible sample has been observed, but no candidate points are inside the estimated feasible region: $\nexists \boldsymbol{x} \in \boldsymbol{E}^{\langle n \rangle}$ such that (9b) holds,

3. The chosen candidate point $\boldsymbol{x}_\theta^{\langle n\rangle}$ fails the expected improvement test (11).

It is enough to show that at least one of these three conditions occurs in finite iterations to prove the Lemma. In particular, in a way similar to [8] (Lemma 2), we can show that this applies to the third case.

At iteration n, consider $\mathscr{E}^{\langle n\rangle} \subset \mathscr{X}$ as the region eligible for an exploitation, i.e.,

$$\mathscr{E}^{\langle n\rangle} \triangleq \left\{\boldsymbol{x} \in \mathscr{X} : \underline{f}^{\langle n\rangle}(\boldsymbol{x}) < z^{*\langle n\rangle} - \alpha\tilde{\gamma}^{\langle n\rangle}\right\}.$$

Whenever any $\boldsymbol{x}^{\langle n+1\rangle} \in \mathscr{E}^{\langle n\rangle}$ passes the test (11) and is thus sampled at iteration $n+1$, there are three possible situations:

1. $\exists s \in [1, S] : c_s^{\langle n+1\rangle} < 0$ (the new sample turns out to be unfeasible). Due to the violation of at least one constraint g_s, there is a region $\mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_s\right)$, which is a ball around $\boldsymbol{x}^{\langle n+1\rangle}$ with radius

$$r_s \triangleq \frac{\left|c_s^{\langle n+1\rangle}\right|}{\tilde{\rho}^{\langle n+1\rangle}},$$

such that $\forall \boldsymbol{x} \in \mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_s\right), \tilde{g}_s(\boldsymbol{x}) < 0$, implying estimated violation w.r.t. g_s and the exclusion from our eligible region for exploitation, i.e.

$$\mathscr{E}^{\langle n+1\rangle} = \mathscr{E}^{\langle n\rangle} \setminus \mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_s\right).$$

This means that a region at least the size of the ball is removed from $\mathscr{E}$, which is valid even when $\tilde{\rho}_s^{\langle n\rangle}$ is updated (with an increase or decrease), because $\underline{\rho}_s \leqslant \tilde{\rho}_s^{\langle n\rangle} \leqslant \rho_s$.

2. $z^{\langle n+1\rangle} \geqslant z^{*\langle n\rangle}$ (the new sample does not improve over the best one). In this case, we consider $\mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_\theta\right)$ with

$$r_\theta \triangleq \frac{z^{\langle n+1\rangle} - \left(z^{*\langle n\rangle} - \alpha\tilde{\gamma}^{\langle n\rangle}\right)}{\tilde{\gamma}^{\langle n+1\rangle}}.$$

Then $\forall \boldsymbol{x} \in \mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_\theta\right)$ it holds that $\underline{f}^{\langle n+1\rangle}(\boldsymbol{x}) \geqslant z^{*\langle n\rangle} - \alpha\tilde{\gamma}^{\langle n\rangle}$. Therefore all the points inside the hyper-ball are not eligible for exploitation at iteration $n+1$. Hence, we have $\mathscr{E}^{\langle n+1\rangle} = \mathscr{E}^{\langle n\rangle} \setminus \mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_\theta\right)$, thus again reducing the volume of the set of candidates that are eligible for exploitation.

3. $z^{\langle n+1\rangle} < z^{*\langle n\rangle}$ (the new sample becomes the best one). In this case, the set of candidate points for exploitation shrinks, i.e. $\mathscr{E}^{\langle n+1\rangle} \subset \mathscr{E}^{\langle n\rangle}$ due to the new threshold. Moreover, the hyper-ball $\mathscr{B}\left(\boldsymbol{x}^{\langle n+1\rangle}, r_\theta\right)$, with $r_\theta = \alpha$ around the new sample is also removed from $\mathscr{E}$.

Since in all the cases the volume of the set $\mathscr{E}^{\langle n\rangle}$ decreases by a finite quantity, we have that $\mathscr{E}^{\langle n\rangle} = \varnothing$ after finite iterations, and exploitation will fail, proving the lemma. □

The following lemma, which is essential in building our convergence theorem, proves that the exploration will generate an increasingly dense distribution of points throughout the search space.

Lemma 2. *Assume that exploration is called infinitely often as $n \to \infty$. Then, for any $\hat{\boldsymbol{x}} \in \mathscr{X}$ and any $\sigma > 0, \exists n_\sigma < \infty$ such that*

$$\min_{\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n_\sigma\rangle}} \|\boldsymbol{x}^{(i)} - \hat{\boldsymbol{x}}\| < \sigma.$$

Proof. Consider any point $\hat{\boldsymbol{x}} \in \mathscr{X} \setminus \boldsymbol{X}^{\langle n\rangle}$ and its nearest sample

$$\overline{\boldsymbol{x}} = \operatorname{argmin}_{\boldsymbol{x} \in \boldsymbol{X}^{\langle n\rangle}} \|\boldsymbol{x} - \hat{\boldsymbol{x}}\|, \tag{27}$$

and define

$$\mathscr{Q} \triangleq \left\{\boldsymbol{x} \in \mathscr{X} | (\boldsymbol{x} - \overline{\boldsymbol{x}})^\top (\hat{\boldsymbol{x}} - \overline{\boldsymbol{x}}) > 0\right\}. \tag{28}$$

Moreover, consider the point

$$\tilde{\boldsymbol{x}} \triangleq \begin{cases} \operatorname{argmin}_{\boldsymbol{x} \in \mathscr{Q} \cap \boldsymbol{X}^{\langle n\rangle}} \|\boldsymbol{x} - \overline{\boldsymbol{x}}\| & \text{if } \mathscr{Q} \cap \boldsymbol{X}^{\langle n\rangle} \neq \varnothing, \\ \operatorname{argmax}_{\boldsymbol{x} \in \mathscr{Q}} \|\boldsymbol{x} - \overline{\boldsymbol{x}}\| & \text{otherwise.} \end{cases} \tag{29}$$

By construction, we have an empty half-ball

$$\mathscr{H} = \mathscr{Q} \cap \mathscr{B}(\overline{\boldsymbol{x}}, \|\overline{\boldsymbol{x}} - \tilde{\boldsymbol{x}}\|) \tag{30}$$

and we show that there will be a candidate point sampled within $\mathscr{H}$ after finite iterations. Due to the mechanism described in the algorithm, existence of candidate points within $\mathscr{H}$ is guaranteed in both cases for $\tilde{\boldsymbol{x}}$ in (29). In the first case ($\mathscr{Q} \cap \boldsymbol{X}^{\langle n \rangle} \neq \varnothing$), this is due to those along the segment between $\overline{\boldsymbol{x}}$ and $\tilde{\boldsymbol{x}}$. For the other case, candidate points are still guaranteed to exist because of the ones generated in the coordinate directions. Now, we assume for the sake of contradiction that points within $\mathscr{H}$ are never sampled.

We consider a candidate point $\boldsymbol{e} \in \mathscr{H}$. Furthermore, consider the ranking of candidate points in $\boldsymbol{E}^{\langle n \rangle}$, in decreasing order of the merit function $\xi_{\psi}^{\langle n \rangle}(\boldsymbol{x})$:

$$\boldsymbol{M}^{\langle n \rangle} \triangleq \left\{ \boldsymbol{m}_1^{\langle n \rangle}, \boldsymbol{m}_2^{\langle n \rangle}, \boldsymbol{m}_3^{\langle n \rangle}, \ldots \right\}.$$

We recall the definition of $\xi_{\psi}^{\langle n \rangle}(\boldsymbol{x})$ in (13) and the finite-boundedness of $h^{\langle n \rangle}(\boldsymbol{x})$,

$$\underline{h} \leqslant h^{\langle n \rangle}(\boldsymbol{x}) \leqslant \overline{h}.$$

Assume that $\boldsymbol{e}$ is generated at iteration n_g. We first calculate an upper bound on the number of iterations n_e such that from iteration $n_g + n_e$, no candidate points will be generated with merit greater than $\xi_{\psi}(\boldsymbol{e})$. Denoting a candidate point generated at $n_g + n_e$ as $\boldsymbol{e}\prime$, and noting that $\xi_{\psi}^{\langle n_g+n_e \rangle}(\boldsymbol{e}\prime) \leqslant \overline{h}$ (age of $\boldsymbol{e}\prime$ is zero at time of generation) and $\xi_{\psi}^{\langle n_g+n_e \rangle}(\boldsymbol{e}) \geqslant \underline{h} + \psi k(n_e)$, it applies that

$$n_e \leqslant k^{-1}\left(\frac{\overline{h} - \underline{h}}{\psi}\right), \tag{31}$$

where k^{-1} is the inverse function of k. Now we count the number $M_{>}$ of candidate points in $\boldsymbol{M}^{\langle n_g+n_e \rangle}$ with greater merit than $\xi_{\psi}^{\langle n_g+n_e \rangle}(\boldsymbol{e})$. The upper bound for such a number is when all generated candidate points from iteration n_g to $n_g + n_e$ have merits exceeding that of $\boldsymbol{e}$, hence, recalling the candidate points generation mechanism, we have that

$$M_{>} \leqslant \left|\boldsymbol{M}^{\langle n_g \rangle}\right| + (B-1)\sum_{n=n_g}^{n_g+n_e} n\left(2D + \frac{n-1}{2}\right) < \infty.$$

where $|\cdot|$ is the set cardinality operator. This implies that $\boldsymbol{e}$ will emerge in front of the ranking $\boldsymbol{M}\langle n_g + M_{>} \rangle$, and will be taken for exploration after at most $n_g + M_{>}$ iterations, falling into contradiction. After sampling $\boldsymbol{e} \in \mathscr{H}$, $\mathscr{H}$ can be redefined using $\tilde{\boldsymbol{x}} \leftarrow \boldsymbol{e}$, and the same reasoning is repeated to show that the considered $\mathscr{H}$ shrinks, and

$$\lim_{n\to\infty} \|\tilde{\boldsymbol{x}}^{\langle n \rangle} - \bar{\boldsymbol{x}}^{\langle n \rangle}\| = 0. \tag{32}$$

From (32), and using the arguments in Lemma 5 of [8], it applies that

$$\lim_{n\to\infty} \|\bar{\boldsymbol{x}}^{\langle n \rangle} - \hat{\boldsymbol{x}}\| = 0,$$

and, for any $\sigma > 0$, we have

$$\exists n_\sigma : \min_{\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n_\sigma \rangle}} \|\boldsymbol{x}^{(i)} - \hat{\boldsymbol{x}}\| < \sigma,$$

which completes the proof. □

Given the above lemmas, we are now ready to present the main convergence result for SMGO-Δ.

Theorem 1. *Let Assumption 1 hold. Then,* $\forall \varepsilon > 0,\ \exists n_\varepsilon < \infty : z^{*\langle n_\varepsilon \rangle} \leqslant z^* + \varepsilon$.

Proof. Consider a point $\boldsymbol{x}^* \in \{\boldsymbol{x}\prime | \boldsymbol{x}\prime = \operatorname{argmin}_{\boldsymbol{x}\in\mathscr{X}} f(\boldsymbol{x})\}$ and take $\sigma = \frac{\varepsilon}{\gamma}$. For any n, denote

$$\bar{\boldsymbol{x}}^{\langle n \rangle} = \arg\min_{\boldsymbol{x}^{(i)} \in \boldsymbol{X}^{\langle n \rangle}} \|\boldsymbol{x}^* - \boldsymbol{x}^{(i)}\|$$

In virtue of Lemma 1, the exploration mode will be called infinitely often. Now, by applying Lemma 2, we have that $\exists n_\sigma < \infty : \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}^{\langle n_\sigma \rangle}\| < \sigma = \frac{\varepsilon}{\gamma}$. Then, in virtue of Assumption 1 we have:

$$f\left(\boldsymbol{x}^{*\langle n_\sigma \rangle}\right) - f(\boldsymbol{x}^*) = z^{*\langle n \rangle} - z^* \leqslant f\left(\bar{\boldsymbol{x}}^{\langle n_\sigma \rangle}\right) - f(\boldsymbol{x}^*) \leqslant \gamma \|\boldsymbol{x}^* - \bar{\boldsymbol{x}}^{\langle n_\sigma \rangle}\| < \varepsilon$$

Thus proving the result with $n_\varepsilon = n_\sigma$. □

5. Extensions and implementation aspects

5.1. On bounded noise and disturbances

Our previous work [7,8] considered cases in which black-box function evaluations are assumed exact and without noise. However, in practice, the values of the objective (resp. constraint) function can be affected by an additive disturbance ϵ_f (resp. ϵ_s) for any test point $\boldsymbol{x} \in \mathscr{X}$, which we assume to be bounded:

$$\forall n, z^{\langle n \rangle} = f(\boldsymbol{x}^n) + \epsilon_f, |\epsilon_f| \leqslant \overline{\epsilon}_f \quad (33)$$

$$c_s^{\langle n \rangle} = g_s(\boldsymbol{x}^n) + \epsilon_s, |\epsilon_s| \leqslant \overline{\epsilon}_s, s = 1, \ldots, S. \quad (34)$$

In this case, we can derive estimates of the disturbance bounds, $\tilde{\epsilon}_f$ and $\tilde{\epsilon}_s$, from $\boldsymbol{X}^{\langle n \rangle}$, resorting to the method proposed in [43]:

$$\tilde{\epsilon}_f^{\langle n \rangle} = \frac{1}{n} \sum_{i=1}^{n} \epsilon_f^{(i)}, \quad (35)$$

$$\tilde{\epsilon}_s^{\langle n \rangle} = \frac{1}{n} \sum_{i=1}^{n} \epsilon_s^{(i)}, \quad (36)$$

where

$$\epsilon_f^{(i)} = \max_{(\boldsymbol{x},z,\boldsymbol{c}) \in \boldsymbol{J}^{(i)}} |z^{(i)} - z|,$$

$$\epsilon_s^{(i)} = \max_{(\boldsymbol{x},z,\boldsymbol{c}) \in \boldsymbol{J}^{(i)}} |c^{(i)} s - c_s|$$

and

$$\boldsymbol{J}^{(i)} \triangleq \left\{ (\boldsymbol{x}, z, \boldsymbol{c}) \in \boldsymbol{X}^{\langle n \rangle} \,|\, \|\boldsymbol{x}^{(i)} - \boldsymbol{x}\| \leqslant v \right\}$$

for a chosen (small) radius $v > 0$. As a rule of thumb, one can set $v = 0.1 d(\mathscr{X})$, where $d(\mathscr{X})$ is the diameter of the search space. If $\boldsymbol{J}^{(i)} = \varnothing \forall i = 1, \ldots, n$, one can increase v and repeat the calculations. Furthermore, the algorithm now updates the Lipschitz constants' estimates $\tilde{\gamma}^{\langle n \rangle}$ and $\tilde{\rho}_s^{\langle n \rangle}$, $s = 1, \ldots, S$ (see Assumption 1) as in [43]:

$$\tilde{\gamma}^{\langle n \rangle} = \max_{\left(\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}\right), \left(\boldsymbol{x}^{(j)}, z^{(j)}, \boldsymbol{c}^{(j)}\right) \in \boldsymbol{X}^{\langle n \rangle}} \left(\begin{cases} \frac{|z^{(i)} - z^{(j)}| - 2\tilde{\epsilon}_f^{\langle n \rangle}}{\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|} & \text{if } |z^{(i)} - z^{(j)}| \geqslant 2\tilde{\epsilon}_f^{\langle n \rangle} \\ \underline{\gamma} & \text{otherwise} \end{cases} \right), \quad (37)$$

$$\tilde{\rho}_s^{\langle n \rangle} = \max_{\left(\boldsymbol{x}^{(i)}, z^{(i)}, \boldsymbol{c}^{(i)}\right), \left(\boldsymbol{x}^{(j)}, z^{(j)}, \boldsymbol{c}^{(j)}\right) \in \boldsymbol{X}^{\langle n \rangle}} \left(\begin{cases} \frac{|c^{(i)} s - c^{(j)} s| - 2\tilde{\epsilon}_s^{\langle n \rangle}}{\|\boldsymbol{x}^{(i)} - \boldsymbol{x}^{(j)}\|} & \text{if } |c_s^{(i)} - c_s^{(j)}| \geqslant 2\tilde{\epsilon}_s^{\langle n \rangle} \\ \underline{\rho}_s & \text{otherwise} \end{cases} \right). \quad (38)$$

Remark 4. The theoretical results laid out in Lemmas 1 and 2 still hold valid even when considering the case with bounded noise, that is, the exploitation routine will end in finite iterations, and the algorithm will densely cover the search set. In fact, since the update of $\tilde{\epsilon}_f^{\langle n \rangle}$ applies to all sample points $\boldsymbol{x}^{(i)}$ when calculating $\underline{f}^{\langle n \rangle}(\boldsymbol{x})$ (see (7)), any increase or decrease of such an estimate will only shift the lower bounds surface by a constant term. Hence, all results discussed above still hold valid. Furthermore, Lemma 2 is anchored to the candidate points generation and the age-based term in the exploration merit ranking, hence the dense points argument still holds.

5.2. On the choice of search directions

The proposed candidate points generation scheme is simple and reproducible, but it may limit the possible directions to sample throughout the search space. Hence, we introduce, in addition to the candidate points generation routine described in Section 3.4, a pseudo-random points distribution throughout $\mathscr{X}$. As in [44], we use a D-dimensional Sobol sequence to generate such points.

In this case, the set of exploration candidate points $\boldsymbol{E}^{\langle 0 \rangle}$ is initialized as a finite set of L points from the Sobol sequence (in this paper, we set $L = 500$). From the first sample onward, $\boldsymbol{E}^{\langle n \rangle}$ is augmented with the new points as in Section 3.4. Note that these initial candidate points share the properties of all other succeeding ones:

- fixed candidate point locations from the time of generation, and
- increasing age w.r.t. n,

which maintains the validity of all the theoretical properties discussed in Section 4.

A similarly-generated set of L pseudo-random candidate points is also used in exploitation to improve the coverage of the trust region. As $\mathscr{T}^{\langle n \rangle}$ expands or contracts, the pseudo-random points proportionally scale in the same manner.

They not only increase the search directions in the exploitation step, but more importantly guarantee the existence of at least L candidate points inside the thrust region $\mathscr{T}^{\langle n \rangle}$, even if it becomes small. The trust region filler points change locations w.r.t. iterations, depending on the size and location of $\mathscr{T}^{\langle n \rangle}$. Nevertheless, the arguments laid out in Lemma 1 still hold true, because they do not require any assumption on the location of the exploitation candidate points. Furthermore, assuming the same seed in the pseudo-random candidate points generation, we still maintain the reproducibility property of SMGO-Δ.

5.3. On the computational complexity

The computational complexity of the proposed algorithm SMGO-Δ is lower than that of the unconstrained SMGO discussed in [8] regardless of consideration of additive uncertainty. This is mainly because of the candidate points' generation mechanism, which leads now to $\mathcal{O}(Dn + n^2)$, as compared with $\mathcal{O}\left(2^D + n^2\right)$ from [7,8] for a D-dimensional hyperrectangle as search space.

In the noiseless case, the Lipschitz constant update can be done in an iterative manner as in [8]. Similarly, the routines to calculate the (upper- and lower-) bounds and the central estimates of $f(\boldsymbol{x})$ and $g_s(\boldsymbol{x})$ can take an iterative implementation.

When noise on the objective and/or constraints should be considered, the noise amplitudes estimation entails the recalculation of the radius v with which the sample differences are calculated, hence this cannot be done iteratively in general. As the Lipschitz constant calculations depend on the estimated noise amplitudes, these also have to be repeated for each iteration. As both these processes require comparing each sample with all the other ones (and the comparison is linear w.r.t. D), these processes have complexity $\mathcal{O}\left(Dn^2\right)$, which is still polynomial.

The introduction of pseudo-random points as candidates for exploration does not change the complexity of the whole algorithm, since these are treated like all other candidate points in terms of the calculations required. On the other hand, the introduction of pseudo-random points within $\mathscr{T}^{\langle n \rangle}$ for exploitation requires the recalculation of (9a) each time the best point or the size of the thrust region are modified. Nevertheless, the computations needed for a fixed number of points do not impact the complexity.

6. Performance and sensitivity analysis

We now analyze, in an illustrative low-dimensional problem, the behavior of SMGO-Δ, focusing on its sensitivity w.r.t. the user-defined parameters involved. The source code used for the tests are available in the following URL: https://github.com/lorenzosabugjr/smgo-delta.

6.1. Illustrative example

We consider the optimization of a 2D function with 2 constraints, see Tables 2,3, using $\alpha = 0.005$ and $\Delta = 0.25$. Fig. 2 shows the search space with the objective function contours. The shaded regions are the respective feasible regions for g_1 and g_2, resulting in disjoint feasible regions composed of a circle and portions of circular strips. The global minimum is marked with an $\otimes$, while other local minima are shown with $\odot$.

The history of sampled values $z^{\langle n \rangle}$ and the best value $z^{*\langle n \rangle}$ is shown in Fig. 3, along with the sample feasibility and the operation mode taken. The $z^{*\langle n \rangle}$ update is solely done when a better *feasible* sample is found. Hence, there were samples with improved costs but found in violation with one or more constraints, and did not change the previous $z^{*\langle n \rangle}$.

Fig. 4 shows the resulting spread of samples in the search space. Mostly, the samples were acquired within the feasible area, and exhibited a concentration around the global optimal point. This is mainly due to a relatively low improvement threshold factor α, leading to more exploitation points being taken (chosen from candidate points in the estimated feasible region). On the other hand, the exploration iterations were used to sample the candidate points within the feasible region, due to a low Δ parameter, leading to a conservative exploration behavior. However, notice that even with such conservatism, there were still a sizeable number of samples which fell outside the feasible region. This is due to the *a priori* unknown noise bounds and Lipschitz constants for the constraints, both of which have to be estimated from data.

6.2. Influence of risk factor Δ

Now we show the effects of the Δ parameter on the exploration behavior of SMGO-Δ. Since we want to focus on the exploration behavior, we set $\alpha = 100$ to completely suppress the exploitation part of the algorithm. Furthermore, the Δ value is

Table 2
Functions used for illustrative tests.

Description	Function definition	Comments
Objective $f(\boldsymbol{x})$	$\frac{1}{2}\sum_{i=1}^{2}(x_i^4 - 16x_i^2 + 5x_i) + 80$	Styblinski-Tang function with offset (multimodal, separable)
Constraint $g_1(\boldsymbol{x})$	$-4 + \|\boldsymbol{x} - [-2.90 \quad 2.90]^\top\|$	Upwards-facing cone centered at $(-2.90, 2.90)$
Constraint $g_2(\boldsymbol{x})$	$\cos\left(2\|\boldsymbol{x} + [2.90 \quad 2.90]^\top\|\right)$	Concentric ripples around $(-2.90, -2.90)$

Table 3
Test parameters for the illustrative example.

Description	Value
Search space $\mathscr{X}$	$-5 \leqslant x_i \leqslant +5, i = 1, 2$
Starting point $\boldsymbol{x}^{(1)}$	(0.4775, 0.0667)
Maximum iterations N	500
$f(\boldsymbol{x})$ noise amplitude $\bar{\epsilon}_f$	0.25 (white noise)
$g_1(\boldsymbol{x})$ noise amplitude $\bar{\epsilon}_1$	0.1 (white noise)
$g_2(\boldsymbol{x})$ noise amplitude $\bar{\epsilon}_2$	0.05 (white noise)

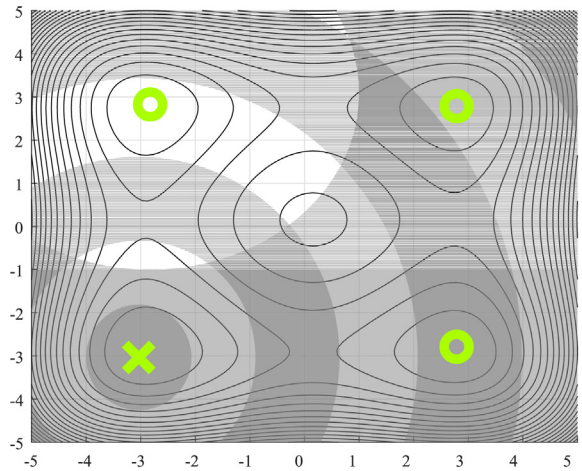

Fig. 2. Illustrative example: contour plot of the objective function, with constraint satisfaction regions.

varied from 0.0 to 1.0, in increments of 0.25, while keeping all other hyperparameters constant. The resulting sample points distributions are shown in Fig. 5. In general, more points are sampled in the unfeasible region when using higher Δ value. However, for too small Δ (below 0.25 in this example), the algorithm is not able to discover part of the feasible region, such as the one containing $\boldsymbol{x}^*$. At the limit of $\Delta = 1.0$, we can observe a quasi-uniform points distribution, however the samples density is slightly higher in the feasible regions compared to those where constraints are violated.

6.3. Sensitivity w.r.t. α and ϕ

Several trends of the algorithm behavior are investigated when the main parameters are varied as follows:

- $\alpha \in \{0.001, 0.0025, 0.005, 0.01, 0.025, 0.05\}$
- $\Delta \in \{0.0, 0.1, 0.2, \ldots, 0.9, 1.0\}$

Each combination (α_i, Δ_j) is tested by running 100 independent tests of the problem described by Tables 2,3. The starting points across these 100 runs (for the same (α_i, Δ_j) combination) are randomly generated. However, the set of starting points are the same across different (α_i, Δ_j) combinations to obtain a fair comparison.

Fig. 6 shows the contour graph of the average optimality gap, measured as $z^{*\langle n \rangle} - z^*$, for the different combinations of α and Δ. Fig. 7 presents the contour of the average percentages of exploitation points taken throughout the runs. On the other hand, Fig. 8 shows the (average) total percent of unfeasible points sampled by the algorithm.

As expected, the percentage of exploitation points decreases w.r.t. increasing α, due to a higher tendency to switch to exploration; this trend is verified by the mostly horizontal contours in Fig. 7. Less-expected observations can be taken from

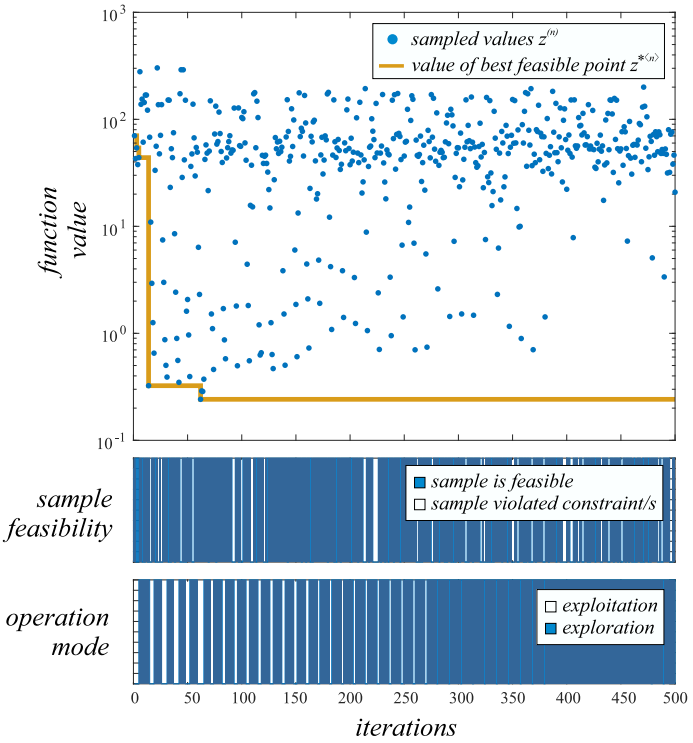

Fig. 3. Illustrative example: sampled values history, best value history, test point feasibility, and SMGO-Δ mode taken (exploitation or exploration).

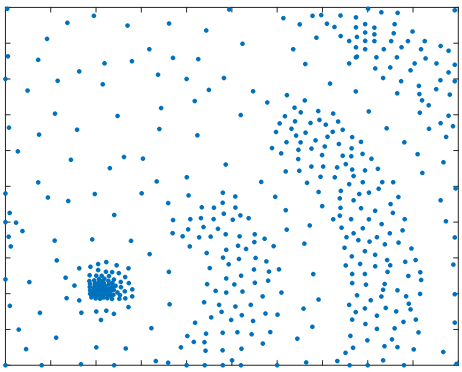

Fig. 4. Illustrative example: resulting sample points distribution.

Fig. 8, regarding the relation of α and Δ to the "cautiousness" of the algorithm. In a general case, we see in Fig. 8 that the percentage of unfeasible samples increases with the increment of both α and Δ. Also we can see that there are two approaches to increase cautiousness: to decrease α (exploit more), or to decrease Δ, as evident in the vertical contours on the left side, and the horizontal contours on the bottom part of Fig. 8.

True enough, adjusting Δ to larger values has its own incentives and trade-offs. As shown in Fig. 6, the optimality gap w.r.t. $\boldsymbol{x}^*$ improves with higher Δ, especially with relatively large values of α. However, as discussed, this is at the cost of increasing the unfeasible samples count. This trade-off introduces a perspective of balancing *risk* and *reward*, which can be tuned by the user. For this particular case, the use of a riskier exploration (higher Δ parameter) had its incentives. This is because as shown in Fig. 4, the feasible regions are disjoint, and the two local minima (one of which is the global one) are located in separate and faraway regions, rewarding a more aggressive exploration. In some cases, we can decide to have a more conservative exploration (choose a smaller Δ) when given a relatively small iteration budget w.r.t. search space, e.g. when the problem is relatively high dimensional, or when large constraint violations shall be avoided. However, as pointed out in Remark 2, a completely safe optimization is not guaranteed in general even when $\Delta \to 0.0$, due to the lack of knowledge about the true Lipschitz constants ρ_s, which holds for a majority of contexts.

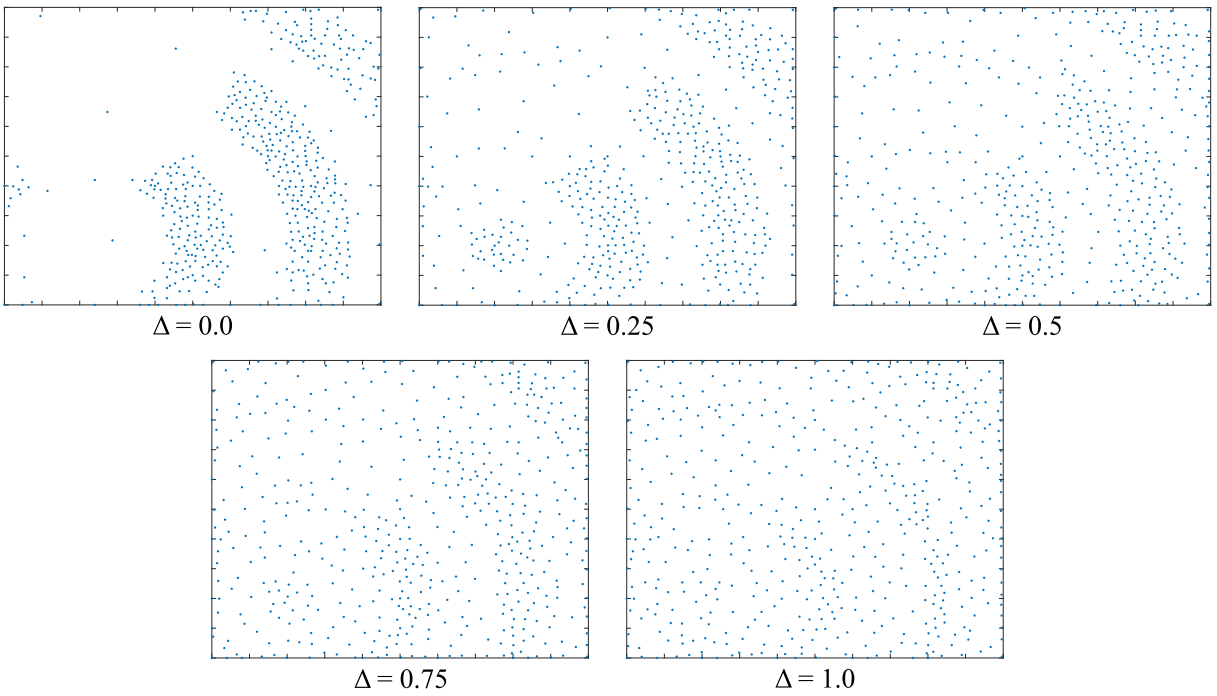

Fig. 5. Illustrative example: results of varying the risk factor Δ.

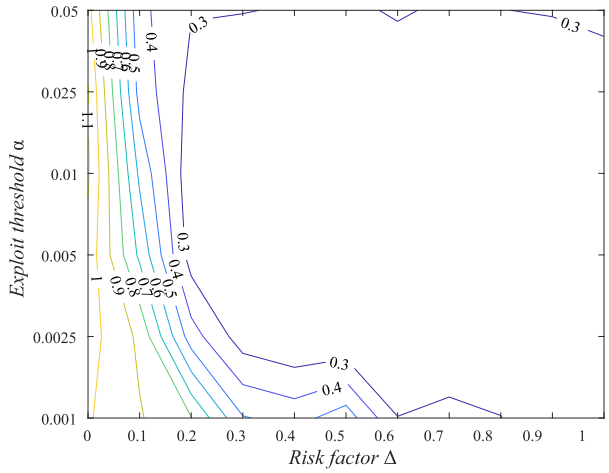

Fig. 6. Illustrative example: contour plots of optimality gap, variations w.r.t. α and Δ.

In choosing a good combination of α and Δ for our next experiments, we recognize in Fig. 6 that the mean of the optimality gap improves from the bottom left corner going to the top right, but there has been no significant improvement in the region where $\alpha > 0.005$ and $\Delta > 0.20$. However in the corresponding area in Fig. 8, increasing α and Δ is only paid by more unfeasible samples, which is not a reasonable trade-off. Hence, based on the results for this particular problem, we choose the combination of $\alpha = 0.005$ and $\Delta = 0.20$ for the studies presented next. We now summarize the resulting SMGO-Δ hyperparameters in Table 4, used in the succeeding tests, and recommended as default values.

7. Benchmarks

We now compare SMGO-Δ with representative existing methods for black-box optimization with black-box constraints. The first method is the constrained Bayesian optimizer (CBO) as available in MATLAB Optimization Toolbox (R2021a), the most common approach for black-box optimization in the recent literature. Another competitor method we used is the Non-

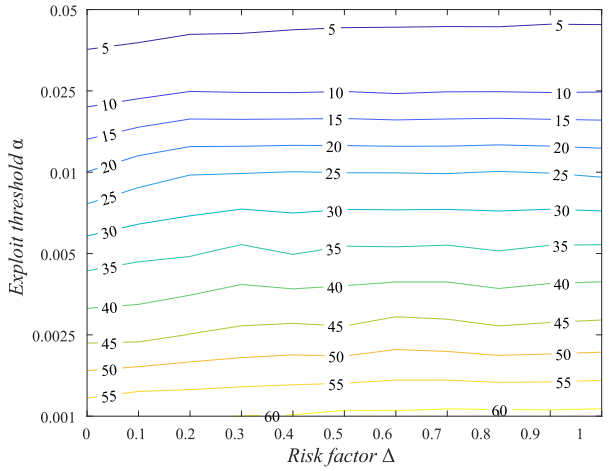

Fig. 7. Illustrative example: average percentage of generated exploitation points, variations w.r.t. α and Δ.

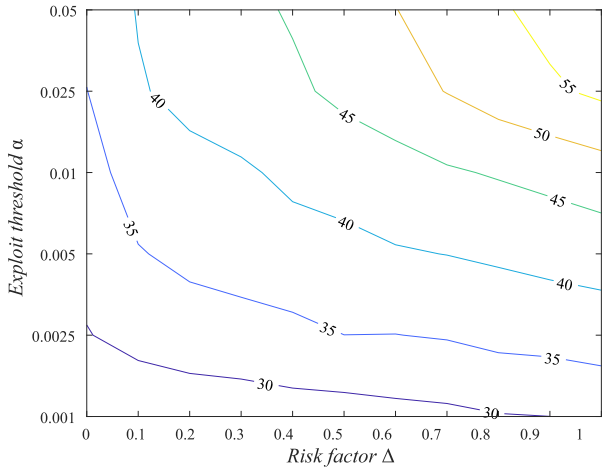

Fig. 8. Illustrative example: average percentage of unfeasible samples, variations w.r.t. α and Δ.

linear Optimization by Mesh Adaptive Direct Search (NOMAD) [19,20], another widely-used algorithm, of which the latest version NOMAD 4 [20] is made available by its authors. We use the default settings for the competitor algorithms, while the settings for SMGO-Δ are as chosen in the last section.

We subject all test algorithms to a benchmark with 10 synthetic problems, with different properties of the objective and constraint functions. For each problem and test algorithm, we performed a statistical test comprising of 50 independent runs, each run having 500 function evaluations[3]. To preserve fairness of comparison, all test algorithms for the same problem use the same set of starting points for the respective independent runs.

7.1. Benchmark problems

A summarized overview of the synthetic problems we consider are given in Table 5. The first 7 problems are from the CEC2006 benchmark set [45], and are commonly-used in benchmarking black-box optimization methods. Furthermore, T1 is a test problem used in [15,33,34], while T2 and T3 are from [15,30]. More details on the function definitions are provided in the Appendix. Note that some inequality constraints are enumerated in the form $g(\boldsymbol{x}) \leqslant 0$ to keep consistent with their respective source literature, but these have been appropriately adjusted to follow $g(\boldsymbol{x}) \geqslant 0$ when used with SMGO-Δ.

Two problems denoted as G05MOD and G23MOD are modified versions of G05 and G23 found in [45]. G05MOD takes from G05, but its equality constraints are converted to inequality constraints. G23MOD, on the other hand, is based on G23 [45] but the equality constraints are removed.

[3] In this section, we refer to the iterations as *function evaluations* to be consistent with the MADS literature.

Table 4
Default SMGO-Δ hyperparameters summary.

Parameter	Description	Default value
α	Expected improvement threshold factor	0.005
Δ	Risk parameter	0.20
β	Weighting factor between $\tilde{f}^{\langle n \rangle}(\boldsymbol{x})$ and $\lambda^{\langle n \rangle}(\boldsymbol{x})$ for exploitation cost (see Section 3.2)	0.1
ϕ	Candidate point age-based merit growth rate	1×10^{-6}
B	Multiplier for generated candidate points per iteration (see Section 3.4)	5
L	Number of generated pseudo-random candidate points for exploitation and resp. exploration	500
$\overline{v}$	Trust region maximum side measure	0.1
κ	Shrink factor for the trust region	0.5
$\underline{v}$	Trust region minimum side measure	$\kappa^{10}\overline{v}$

Table 5
Summary of benchmark problems. ρ stands for the estimated feasible region coverage w.r.t. search space. L: linear, Q: quadratic, C: cubic, NL: nonlinear. N/A: quantities are not specified in source literature.

Problem	D	S	f type	g type	z^*	ρ
G04	5	6	Q	NL	−3.0665e + 04	52.123%
G05MOD	4	5	C	L + NL	5.1265e + 03	N/A
G08	2	2	NL	NL	−0.0958	0.8560%
G09	7	4	NL	NL	680.6301	0.5121%
G12	3	1	Q	NL	−1.0	4.7713%
G23MOD	9	2	L	NL	N/A	N/A
G24	2	2	L	NL	−5.5080	79.6556%
T1	2	2	L	Q + NL	N/A	N/A
T2	2	1	NL	NL	N/A	N/A
T3	2	1	NL	NL	N/A	N/A

We have selected the benchmark problems to evaluate the performance of the compared optimizers in different dimensionality D, feasible region coverage ρ, as well as characteristics of the objective and constraint/s. Some functions, like G04 and G24 have fairly large feasible region coverage, and while G08 and G09 have coverage of less than 1%. Other functions did not have estimated ρ as available in their respective source literature.

7.2. Graphical comparison

Figs. 9–11 show the best points achieved by CBO, NOMAD 4, and SMGO-Δ from the different runs. In these plots, the greyed areas are the unfeasible regions, and the boundaries of the feasible region are drawn in solid black line. We see that all the results of CBO and SMGO-Δ lie around the same region in the search space. Furthermore, the spread of the SMGO-Δ

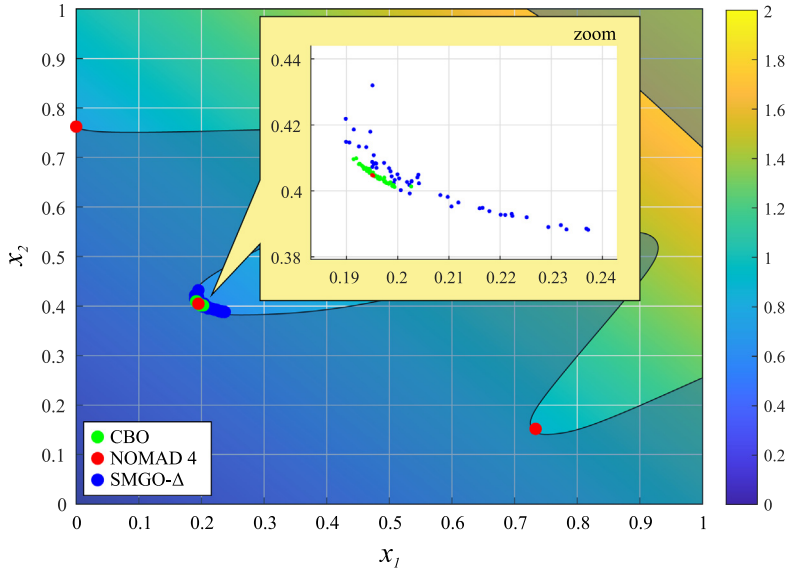

Fig. 9. Benchmark tests: distribution of best results for different optimization runs on T1. The feasible set is shown as the colored region not covered in gray.

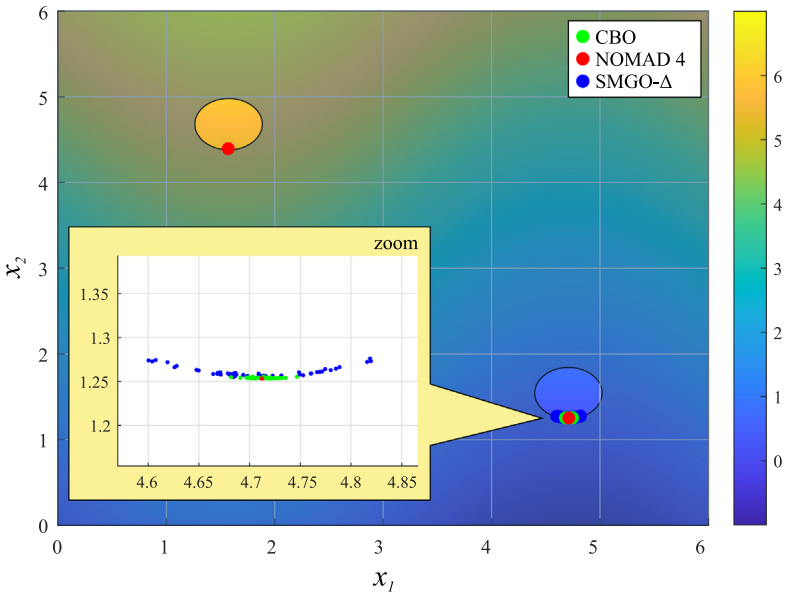

Fig. 10. Benchmark tests: distribution of best results for different optimization runs on T2. The feasible set is composed of two disjoint ellipse-shaped regions.

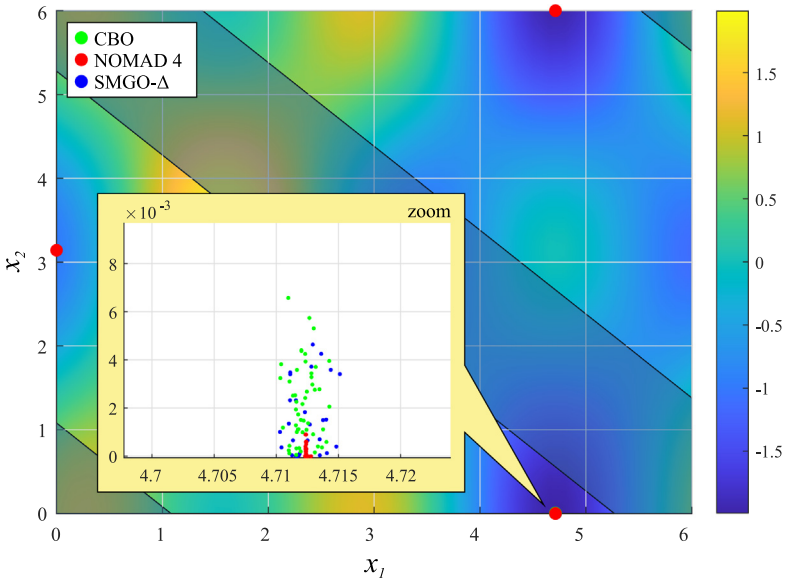

Fig. 11. Benchmark tests: distribution of best results for different optimization runs on T3. The feasible set is shown as the colored stripes not covered in gray.

best points is higher than that of CBO, as seen in the zoom inset of all plots. In the same insets, NOMAD 4 is seen as having the most concentrated distribution, but there are also best points in all plots which lie on a faraway area in the search space. For example, for T2 (Fig. 10), the NOMAD 4 points on the upper left oblong means that at some runs, the feasible region on the lower right (containing the optimal point) is not found. On another case, some NOMAD 4 results for T3 lie on the local minima (center left side, and upper right side of Fig. 11).

7.3. Comparative results

We summarize the average results across 50 runs of the respective compared algorithms in Table 6. In problems G04, G05MOD, and G12, we see that all three methods had highly comparable results, hence we did not highlight any best result in their corresponding rows. For most problems, CBO displayed the best results, for which it resulted in joint best average in

Table 6
Benchmark tests: average results after 500 function evaluations. Best results are highlighted in yellow. $*$: prematurely finished optimization because of maximum gridding refinement, $**$: stopped with no results in 5 separate runs because of no feasible point found after around $n = 90$.

Problem	CBO	NOMAD 4	SMGO-Δ
G04	−3.0518e + 04	−3.0665e + 04	−3.0343e + 04
G05MOD	5.2185e + 03	5.2073e + 03	5.4014e + 03
G08	−0.0958	−0.0823	−0.0958
G09	754.1200	717.6586	1.5131e + 03
G12	−0.9998	−1.000	−0.9671
G23MOD	−3.3832e + 03	−3.9000e + 03	−3.9941e + 03
G24	−5.4262	−5.3852	−5.2789
T1	0.6005	0.6769$*$	0.6088
T2	0.2542	1.9671$*$, $**$	0.2628
T3	−2.0000	−1.8904$*$	−2.0000

Table 7
Benchmark tests: average number of function evaluations before finding the first feasible point. Only the runs starting from an unfeasible point are considered. Best results are highlighted in yellow. $**$: stopped with no results in 5 separate runs because of no feasible point was found after around $n = 90$.

Problem	CBO	NOMAD 4	SMGO-Δ
G04	4.250	19.531	4.938
G05MOD	7.760	66.156	166.540
G08	6.440	18.880	27.860
G09	13.820	62.160	42.020
G12	13.000	29.087	25.500
G23MOD	8.735	46.837	2.449
G24	3.121	6.485	2.667
T1	3.192	9.192	3.192
T2	8.694	20.227$**$	24.102
T3	2.667	4.800	6.133

6 problems. For the others, it had comparable results except for G23MOD where CBO resulted in significantly worse average. NOMAD 4 had 3 joint best averages, and for problems T1-T3, it prematurely finished optimization because of maximum gridding, at around $n = 200$ instead of the maximum $N = 500$. This has led to a worst result for T2, which was 8 times larger than the results for CBO and SMGO-Δ. SMGO-Δ produced competitive results compared with the other algorithms. In 5 problems, it was joint best with either CBO or NOMAD 4. However, its average results with G09 was around 2 times worse than the others, which can be attributed to the problem having relatively higher dimensions, combined with low ρ.

Table 7 shows the average number of function evaluations at the first feasible sample for the respective problems. CBO achieved the most sole/joint best results across all problems, implying its effectiveness in terms of finding a feasible point. SMGO-Δ had the best result for G23MOD, and joint best with CBO on three other problems. However, in problem G05MOD, it was particularly difficult for SMGO-Δ to find a feasible point, resulting in more than twice evaluations as NOMAD 4. This can be associated with a fairly high number of constraints, for which different region/partitions in the search space have different combinations of (predicted) satisfied constraints. This means that in effect, a large region in the search space could have the same number of constraints satisfied, albeit in different combinations. By virtue of the exploration, which prioritizes regions based on number of constraints satisfied, it has taken many evaluations before actually sampling a point that satisfies all 5 constraints. Lastly, NOMAD 4 had no best results in this metric, and with a significant worst result with G23MOD, taking 5 times longer than CBO and 20 times longer than SMGO-Δ before finding a first feasible sample. Furthermore, we point out that although NOMAD 4 had competitive results with problem T2, there were 5 runs in which NOMAD 4 terminated without any result because there was still no feasible point found after around $n = 90$.

We have statistically compared the results of SMGO-Δ with the other algorithms using Wilcoxon and Kruskal–Wallis non-parametric tests, both with 5% significance levels and a null hypothesis that SMGO-Δ results are similar to those from the other methods. We found out that SMGO-Δ distributions for the final results after $N = 500$, as well as for the function evaluations for first feasible sampling, are not statistically similar to those of the other techniques.

8. Case study: MPC tuning for a DC motor

We finally use SMGO-Δ to tune a model predictive controller for a DC motor with model uncertainty and task-level constraints. We compare its performance with an ideal optimal controller with complete information about the actual system model at hand, optimizing the task-level costs considering the entire task duration. Furthermore, we also make a comparison with CBO.

8.1. System model and controller structure

The servo positioning system is shown in Fig. 12, where V is the input voltage, θ_i and θ_o are the input and output shaft angular positions, T is the torsional moment between the input and output shafts, and d is the load (disturbance) on the output shaft. The nameplate nominal values for the motor parameters are given in Table 8.

Assuming linearity of all the components, the system dynamics are described by the following state-space equations (the continuous time variable is omitted for brevity):

$$\dot{\boldsymbol{\xi}} = \begin{bmatrix} 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \\ \frac{-K_\theta}{J_i\tau_g^2} & \frac{-K_\theta}{J_i\tau_g} & -\frac{\beta_i+\frac{K_t^2}{R_m}}{J_i} & 0 \\ \frac{K_\theta}{J_o\tau_g} & \frac{-K_\theta}{J_o} & 0 & \frac{-\beta_o}{J_o} \end{bmatrix} \boldsymbol{\xi} + \begin{bmatrix} 0 \\ 0 \\ \frac{K_t}{J_i R_m} \\ 0 \end{bmatrix} u + \begin{bmatrix} 0 \\ 0 \\ 0 \\ \frac{1}{J_o} \end{bmatrix} d, \tag{39}$$

$$\boldsymbol{y} = \begin{bmatrix} \frac{K_\theta}{\tau_g} & -K_\theta & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \boldsymbol{\xi} \tag{40}$$

where $\boldsymbol{\xi} = \begin{bmatrix} \theta_i\, \theta_o\, \dot{\theta}_i\, \dot{\theta}_o \end{bmatrix}^\top$ is the state, and $\boldsymbol{y} = \begin{bmatrix} T\, \theta_o\, \dot{\theta}_i\, \dot{\theta}_o \end{bmatrix}^\top$ the output.

For the considered system, we design a model predictive controller (MPC, [46]) to track a reference output shaft angle $\hat{\theta}_o$. An MPC strategy is chosen in this case study to explicitly consider the input and state constraints, pertaining to the voltage input and the maximum torsional moments experienced by the shafts.

After setting a sampling time $T_s = 0.1$s, at each time step t the following finite horizon optimal control problem is solved:

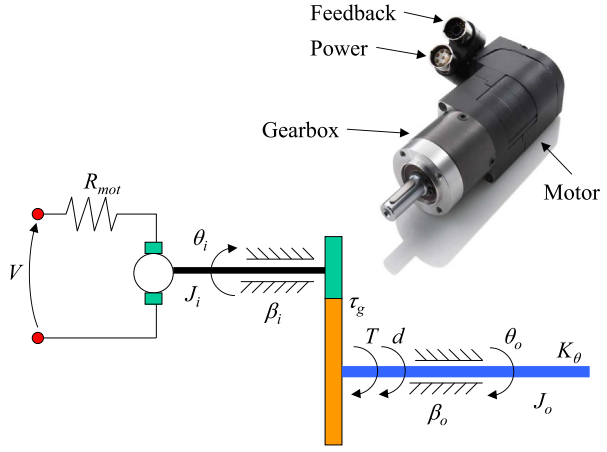

Fig. 12. Case study: servomotor model considered in the MPC tuning problem.

Table 8
Case study: servo motor specifications.

Variable	Description	Nominal value
R_m	Motor electrical resistance	20 Ω
K_t	Motor constant	10 $\frac{\text{Nm}}{\text{A}}$
K_θ	Output shaft torsional stiffness	1280 $\frac{\text{Nm}}{\text{rad}}$
J_i	Input shaft moment of inertia	0.5 kg m^2
J_o	Output shaft moment of inertia	25 kg m^2
β_i	Input shaft friction coefficient	0.1 $\frac{\text{Nms}}{\text{rad}}$
β_o	Output shaft friction coefficient	25 $\frac{\text{Nms}}{\text{rad}}$
τ_g	Gear ratio (input/output)	20

$$\min_{U\in\mathbb{R}^{N_h}} \sum_{i=0}^{N_h} \left(y_{ref} - y(i|t)\right)^\top \mathbf{Q}\left(y_{ref} - y(i|t)\right) + \sum_{i=0}^{N_h-1} Ru(i|t)^2 \tag{41}$$

$$\text{s.t. } \xi(i+1|t) = \boldsymbol{A}\xi(i|t) + Bu(i|t) + B_d d(i|t) \tag{42}$$
$$y(i|t) = \boldsymbol{C}\boldsymbol{x}(i|t) \tag{43}$$
$$\xi(0|t) = \xi(t) \tag{44}$$
$$|u(i|t)| \leqslant \overline{V} \tag{45}$$
$$|T(i|t)| \leqslant \overline{T}, \tag{46}$$

where for each signal, the time instant $(i|t)$ indicates the value of the signal at time $t+i$ predicted at time t, (42) is the discrete-time equivalent of (39) considering a sampling time of 0.1 s, (43) is the output equation given in (40), (44) is the known system state at time step t, (45) declares the input constraint with $\overline{V} = 220$ V, and (46) limits the torsional moment between the gears, with $\overline{T} = 78$ Nm. N_h is the prediction and control horizon, selected as $N_h = 50$. Matrix $\mathbf{Q}$ is a diagonal positive-definite matrix of tunable parameters q_1, q_2, q_3, q_4, while R is fixed as 1.

8.2. Black-box optimization for controller tuning

In the context of factory automation, we encounter machines performing repetitive tasks that usually involve a fixed reference trajectory $\hat{\theta}_o$ and load profile d. Fig. 13 shows the (idealized) task cycle profile we consider, having a duration of $t_{cyc} = 50$ s.

Once the controller architecture is defined, we aim to tune the MPC parameters to guarantee good tracking performance during the complete task, considering the following practically-motivated issues:

1. MPC cost function matrix $\mathbf{Q}$ needs to be tuned to maximize tracking performance on $\hat{\theta}_o$. This concern can usually be treated in the design phase without using black box optimization.
2. The nameplate motor parameters are known (see Table 8); however, the real ones differ due to manufacturing tolerances and/or previous usage. In this paper, $R_m, K_t, \beta_i, \beta_o$ may differ up to $\pm 12.5\%$ w.r.t. nameplate values (the remaining parameters are assumed exact). This introduces model mismatch, possibly leading to degraded MPC performance, and/or violation of constraints (especially (46)). Hence, simultaneous model learning and MPC gain tuning is an appropriate approach for this application.
3. To protect the motor windings and prolong its service lifetime, the average power consumption of the system throughout the task cycle, must not exceed a maximum $\overline{P}$, set as 25 W. At the same time, constraints (45) and (46) must be satisfied for the complete task duration. This declares a constraint affecting the whole task duration, i.e. longer than the MPC prediction horizon. Hence, the satisfaction of this constraint cannot be imposed *a priori* in the MPC design phase.
4. To minimize the cumulative damage to the input–output shaft gear coupling, we limit the total time that the torsional moment T exceeds the limit $\overline{T}$, during the task duration, to be at most $\overline{t}_{viol}$, with $\overline{t}_{viol} = 1$ s.

Based on the previous considerations, the following black-box optimization problem is formulated:

$$\min_{\boldsymbol{x}\in\mathcal{X}} \sum_{t=0}^{t_{max}} \left(\theta_o(t;\boldsymbol{x}) - \hat{\theta}_o(t)\right)^2 \tag{47}$$

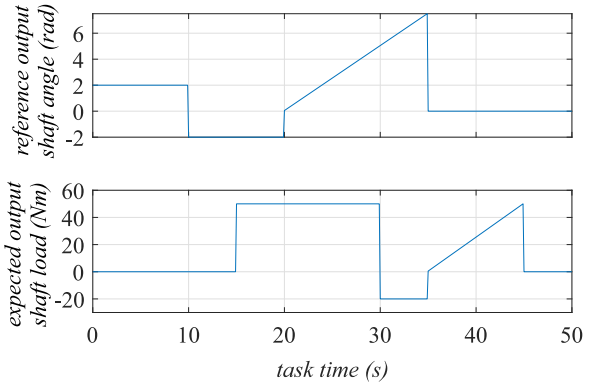

Fig. 13. Case study: reference trajectory and expected load for the servomotor problem.

$$\text{s.t.} \int_{t=0}^{t_{cyc}} \left(\begin{cases} 1 & |T(t)| > \overline{T} \\ 0 & \text{otherwise} \end{cases} \right) dt \leqslant \bar{t}_{viol}, \tag{48}$$

$$\frac{1}{t_{cyc}+1} \sum_{t=0}^{t_{cyc}-1} \frac{u(t)\left(u(t) - K_t \dot{\theta}_i(t)\right)}{R_m} \leqslant \overline{P} \tag{49}$$

where the vector of decision variables is $\boldsymbol{x} = \left[log(q_1)\, log(q_2)\, log(q_3)\, log(q_4)\, \widetilde{R}_m\, \widetilde{K}_t\, \widetilde{\beta}_i\, \widetilde{\beta}_o\right]$. The tilde on 4th to 8th variables of $\boldsymbol{x}$ denotes parameter estimates. $\theta_o(t;\boldsymbol{x})$ is the resulting output shaft angular position at time t, when the MPC controller is applied to the servo system using the parameters $\boldsymbol{x}$. Constraint (48) sets the maximum time of torsional moment violation for the experiment, while (49) imposes the maximum average motor power for the entire task duration.

The search intervals for the first four parameters is $[-7, 7]$, while intervals of $\pm 12.5\%$ w.r.t. nominal values are imposed for the next four (see Table 8 for the nominal values). 25 trials of the SMGO-Δ algorithm are executed with $N = 250$ iterations each, with randomly-generated values of $R_m, K_t, \beta_i, \beta_o$ within the tolerances around the nominal values. For comparison, we also run the constrained Bayesian optimization (CBO, available as `bayesopt` in MATLAB Statistics and Machine Learning Toolbox).

Each trial has the same values of $R_m, K_t, \beta_i, \beta_o$ for both optimizers to guarantee uniformity in the tests. Furthermore, the starting test point $\boldsymbol{x}^{(1)}$ for each trial (both SMGO-Δ and CBO) is composed of $log(q_i) = 0$, and the nominal values for each motor parameter. Each experiment (iteration) is subject to the shaft load as in Fig. 13, but with a different disturbance (2.5 Nm amplitude white noise, band limited to 5 Hz). Again, to maximize uniformity, each iteration has exactly *the same* shaft load disturbance for SMGO-Δ and CBO. That is, the experiments differ from iteration to iteration (due to the load disturbance), but are the same across the tested optimizers.

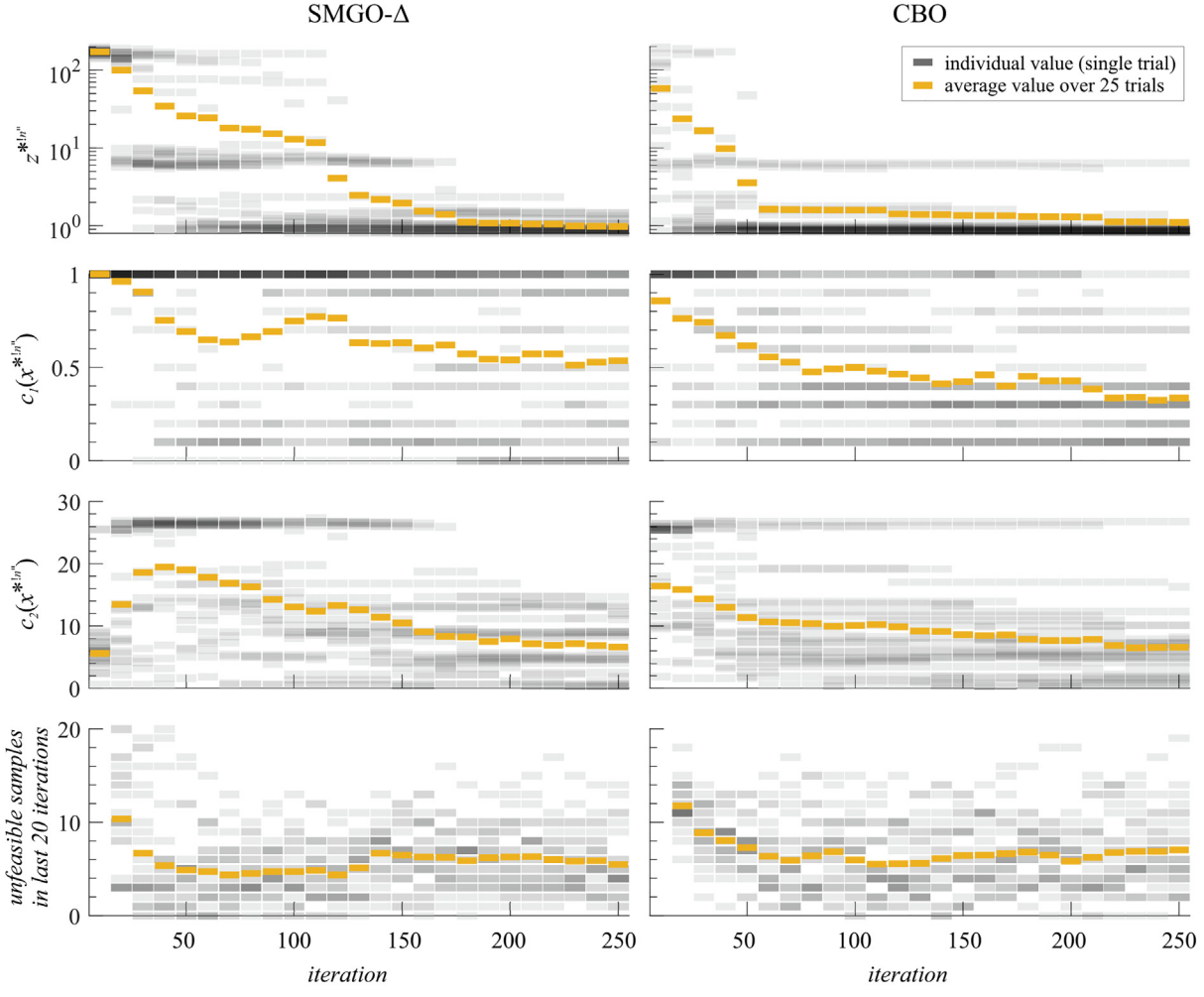

Fig. 14. Case study: MPC tuning results comparison, SMGO-Δ vs CBO: best objective, constraints, and count of unfeasible samples in the previous 20 iterations.

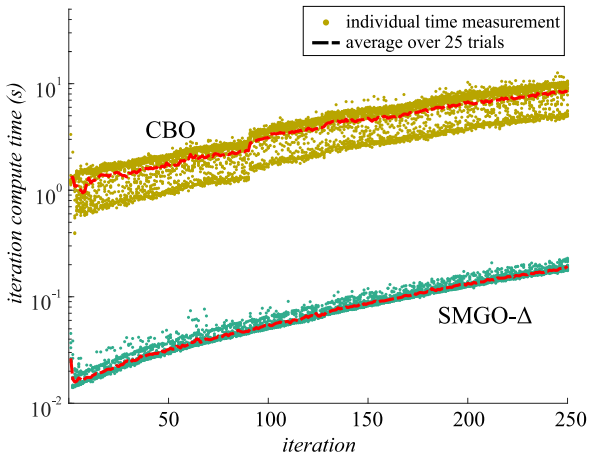

Fig. 15. Case study: computational times comparison, SMGO-Δ vs CBO.

8.3. Comparative results

A comparison of the results from SMGO-Δ and CBO is laid out in Fig. 14. The results show the distributions of best samples history (first row), the corresponding constraint values for the best sample (second and third rows), and the count of unfeasible samples within the last 20 iterations (last row). The individual values from the 25 independent trials, as well as the mean values, are shown for both algorithms.

The iteration-based optimization performance of SMGO-Δ was slower than CBO for this particular test problem, particularly around iterations 30 to 110, where SMGO-Δ suffered from slow improvement of best feasible point, while CBO was already in the range of 1.10 (average) best objective. However, from around $n = 170$, SMGO-Δ managed to produce better average results than CBO, and takes this advantage until the allotted 250 iterations. On the other hand, there were several trials for CBO whose best results were stuck at around 7.0 even after $n = 200$. At the end of the iteration budget, we have obtained an average best objective of 0.979 for SMGO-Δ, and 1.105 for CBO.

The distribution histories for the constraints (for the best sample $\boldsymbol{x}^{*\langle n\rangle}$) show that the best sample is found more and more towards the boundary of feasible set $\mathscr{G}$. This is a reasonable trend, because demanding better tracking performance also means leaning on more aggressive controllers, which in turn are more likely to violate maximum torsional moment and power limits.

The count of unfeasible samples are lower with SMGO-Δ than with CBO, as seen in the last row of Fig. 14. We see that the slow improvement on SMGO-Δ around $n = 30$ to $n = 110$ coincide with the low count of unfeasible samples, which can be because the higher tendency of exploitation at these iterations. However, on the later part of the runs, the average unfeasible samples count for SMGO-Δ and CBO are similar. After the allotted 250 iterations, the average total unfeasible samples of SMGO-Δ (29.6%) are less than that of CBO (34.7%).

A clear advantage can be seen when comparing the computational times of the two algorithms (not including the objective and constraints evaluation times), shown in Fig. 15. It is clear that the proposed SMGO-Δ is much faster than Bayesian, with per-iteration computation times almost 2 orders of magnitude faster. The (average) total time for SMGO-Δ in the entire optimization run (250 iterations) is found to be 20.1 s, which is 51 times shorter than CBO (with 1030 s). Given the much lighter computational burden, and a competitive iteration-based optimization performance, SMGO-Δ can be argued to be a worthy alternative to CBO.

9. Conclusions

A global optimization algorithm based on the Set Membership framework, named SMGO-Δ, is designed for cases where the objective as well as the constraints are black-box, e.g. both can only be evaluated through sampling, possibly affected by bounded noise. The exploitation and exploration strategies of SMGO-Δ are formulated based on the calculation of guaranteed bounds on the cost function and constraints, according to the Set Membership theory. The convergence properties of the proposed algorithm are proven in a theoretical analysis under a Lipschitz continuity assumption of the cost and constraint functions. Furthermore, extensions to noisy samples, enrichment of search directions, and computational cost are discussed. The designed algorithm features good iteration-based performance, repeatability, low computational complexity, and intuitive tunable risk parameter.

The performance and the sensitivity of the algorithm to its user-defined parameters are investigated with an illustrative example function. The SMGO-Δ is compared with state-of-the-art methods in academic benchmarks, showing competitive iteration-based performance. Furthermore, SMGO-Δ has been compared side-by-side with constrained Bayesian optimization in the engineering design context of MPC tuning for an electromechanical system with uncertain parameters. The tests

show the highly comparable performance of SMGO-Δ, with the additional advantage of being around 50 times faster than constrained Bayesian optimization for the treated test case.

CRediT authorship contribution statement

Lorenzo Sabug Jr.: Conceptualization, Methodology, Software, Validation, Formal analysis, Writing – original draft, Visualization. **Fredy Ruiz:** Conceptualization, Supervision, Writing – review & editing. **Lorenzo Fagiano:** Conceptualization, Supervision, Writing - review & editing.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Appendix A. Problem definitions for Benchmark tests

G04 [45]

Description	Value
Objective	$f(\boldsymbol{x}) = 5.3578547x_3^2 + 0.8356891x_1x_5 + 37.293239x_1 - 40792.141$
Constraints	$g_1(\boldsymbol{x}) = 85.334407 + 0.0056858x_2x_5 + 0.0006262x_1x_4 - 0.0022053x_3x_5 - 92 \leqslant 0$
	$g_2(\boldsymbol{x}) = -85.334407 - 0.0056858x_2x_5 - 0.0006262x_1x_4 + 0.0022053x_3x_5 \leqslant 0$
	$g_3(\boldsymbol{x}) = 80.51249 + 0.0071317x_2x_5 + 0.0029955x_1x_2 + 0.0021813x_3^2 - 110 \leqslant 0$
	$g_4(\boldsymbol{x}) = -80.51249 - 0.0071317x_2x_5 - 0.0029955x_1x_2 - 0.0021813x_3^2 + 90 \leqslant 0$
	$g_5(\boldsymbol{x}) = 9.300961 + 0.0047026x_3x_5 + 0.0012547x_1x_3 + 0.0019085x_3x_4 - 25 \leqslant 0$
	$g_6(\boldsymbol{x}) = -9.300961 - 0.0047026x_3x_5 - 0.0012547x_1x_3 - 0.0019085x_3x_4 + 20 \leqslant 0$
Search space	$78 \leqslant x_1 \leqslant 102,$
	$33 \leqslant x_2 \leqslant 45,$
	$27 \leqslant x_i \leqslant 45, i = 3, 4, 5$

G05MOD [31,45]

Description	Value
Objective	$f(\boldsymbol{x}) = 3x_1 + 0.000001x_1^3 + 2x_2 + (0.000002/3)x_2^3$
Constraints	$g_1(\boldsymbol{x}) = x_3 - x_4 - 0.55 \leqslant 0$
	$g_2(\boldsymbol{x}) = x_4 - x_3 - 0.55 \leqslant 0$
	$g_3(\boldsymbol{x}) = 1000\sin(-x_3 - 0.25) + 1000\sin(-x_4 - 0.25) + 894.8 - x_1 \leqslant 0$
	$g_4(\boldsymbol{x}) = 1000\sin(x_3 - 0.25) + 1000\sin(x_3 - x_4 - 0.25) + 894.8 - x_2 \leqslant 0$
	$g_5(\boldsymbol{x}) = 1000\sin(x_4 - 0.25) + 1000\sin(x_4 - x_3 - 0.25) + 1294.8 \leqslant 0$
Search space	$0 \leqslant x_i \leqslant 1200, i = 1, 2$
	$-0.55 \leqslant x_j \leqslant 0.55, j = 3, 4$

G08 [45]

Description	Value
Objective	$f(\boldsymbol{x}) = -\frac{\sin^3(2\pi x_1)\sin(2\pi x_2)}{x_1^3(x_1+x_2)}$
Constraints	$g_1(\boldsymbol{x}) = x_1^2 - x_2 + 1 \leqslant 0$
	$g_2(\boldsymbol{x}) = 1 - x_1 + (x_2 - 4)^2 \leqslant 0$
Search space	$0 \leqslant x_i \leqslant 10, i = 1, 2$

G09 [45]

Description	Value
Objective	$f(\boldsymbol{x}) = (x_1 - 10)^2 + 5(x_2 - 12)^2 + x_3^4 + 3(x_4 - 11)^2 + 10x_5^6 + 7x_6^2 + x_7^4 - 4x_6x_7 - 10x_6 - 8x_7$
Constraints	$g_1(\boldsymbol{x}) = -127 + 2x_1^2 + 3x_2^4 + x_3 + 4x_4^2 + 5x_5 \leqslant 0$
	$g_2(\boldsymbol{x}) = -282 + 7x_1 + 3x_2 + 10x_3^2 + x_4 - x_5 \leqslant 0$
	$g_3(\boldsymbol{x}) = -196 + 23x_1 + x_2^2 + 6x_6^2 - 8x_7 \leqslant 0$
	$g_4(\boldsymbol{x}) = 4x_1^2 + x_2^2 - 3x_1x_2 + 2x_3^2 + 5x_6 - 11x_7 \leqslant 0$
Search space	$-10 \leqslant x_i \leqslant 10, i = 1, \ldots, 7$

G12 [45]

Description	Value
Objective	$f(\boldsymbol{x}) = -\left(100 - (x_1 - 5)^2 - (x_2 - 5)^2 - (x_3 - 5)^2\right)/100$
Constraints	$g_1(\boldsymbol{x}) = (x_1 - p)^2 + (x_2 - q)^2 + (x_3 - r)^2 - 0.0625 \leqslant 0, p, q, r = 1, \ldots, 9$
Search space	$0 \leqslant x_i \leqslant 9, i = 1, 2, 3$
Comments	The feasible region is a set of 9^3 disjoint balls with radius of 0.25, laid out as a grid. Search space is modified so that $\boldsymbol{x}^*$ does not lie in the center.

G23MOD [45]

Description	Value
Objective	$f(\boldsymbol{x}) = -9x_5 - 15x_8 + 6x_1 + 16x_2 + 10(x_6 + x_7)$
Constraints	$g_1(\boldsymbol{x}) = x_9x_3 + 0.02x_6 - 0.025x_5 \leqslant 0$
	$g_2(\boldsymbol{x}) = x_9x_4 + 0.02x_7 - 0.015x_8 \leqslant 0$
Search space	$0 \leqslant x_1, x_2, x_6 \leqslant 300$
	$0 \leqslant x_3, x_5, x_7 \leqslant 100$
	$0 \leqslant x_4, x_8 \leqslant 200$
	$0.01 \leqslant x_9 \leqslant 0.03$
Comments	Taken from G23 as in [45], but removed the equality constraints

G24 [45]

Description	Value
Objective	$f(\boldsymbol{x}) = -x_1 - x_2$
Constraints	$g_1(\boldsymbol{x}) = -2x_1^4 + 8x_1^3 - 8x_1^2 + x_2 - 2 \leqslant 0$
	$g_2(\boldsymbol{x}) = -4x_1^4 + 32x_1^3 - 88x_1^2 + 96x_1 + x_2 - 36 \leqslant 0$
Search space	$0 \leqslant x_1 \leqslant 3$
	$0 \leqslant x_2 \leqslant 4$

T1 [15,33,34]

Description	Value
Objective	$f(\boldsymbol{x}) = x_1 + x_2$
Constraints	$g_1(\boldsymbol{x}) = 0.5\sin(2\pi(x_1^2 - 2x_2)) + x_1 + 2x_2 - 1.5 \geqslant 0$
	$g_2(\boldsymbol{x}) = -x_1^2 - x_2^2 + 1.5 \geqslant 0$
Search space	$0 \leqslant x_i \leqslant 1, i = 1, 2$

T2 [15,30]

Description	Value
Objective	$f(\boldsymbol{x}) = \sin(x_1) + x_2$
Constraints	$g_1(\boldsymbol{x}) = \sin(x_1)\sin(x_2) + 0.95 \leqslant 0$
Search space	$0 \leqslant x_i \leqslant 6, i = 1, 2$

T3 [15,30]

Description	Value
Objective	$f(\boldsymbol{x}) = \cos(2x_1)\cos(x_2) + \sin(x_1)$
Constraints	$g_1(\boldsymbol{x}) = \cos(x_1)\cos(x_2) - \sin(x_1)\sin(x_2) - 0.5 \leqslant 0$
Search space	$0 \leqslant x_i \leqslant 6, i = 1, 2$

References

[1] D.R. Jones, M. Schonlau, and W.J. Welch, Efficient Global Optimization of Expensive Black-Box Functions, J. Global Optim. 13 (1998) 455–492. Available: URL: https://link.springer.com/content/pdf/10.1023%2FA%3A1008306431147.pdf..
[2] H.M. Gutmann, A Radial Basis Function Method for Global Optimization, J. Global Optim. 19 (3) (2001) 201–227.
[3] S.Z. Khong, D. Nešic, C. Manzie, and Y. Tan, Multidimensional global extremum seeking via the direct optimisation algorithm, Automatica 49(7) (2013) 1970–1978..
[4] C. Malherbe and N. Vayatis, Global optimization of Lipschitz functions, in: 34th International Conference on Machine Learning, ICML 2017, vol. 5, 2017, pp. 3592–3601..
[5] S. Gao, L. Shi, and Z. Zhang, A peak-over-threshold search method for global optimization, Automatica 89 (2018) 83–91..
[6] A. Bemporad, Global optimization via inverse distance weighting and radial basis functions, Comput. Optim. Appl. 77 (2) (2020) 571–595.
[7] L. Sabug, F. Ruiz, and L. Fagiano, On the use of Set Membership theory for global optimization of black-box functions, in: 2020 59th IEEE Conference on Decision and Control (CDC), vol. 2020-Decem. IEEE, Dec 2020, pp. 3586–3591. Available: URL: https://ieeexplore.ieee.org/document/9304123/..
[8] L. Sabug, F. Ruiz, and L. Fagiano, SMGO: A set membership approach to data-driven global optimization, Automatica 133 (2021) 109890..
[9] M. Urquhart, E. Ljungskog, and S. Sebben, Surrogate-based optimisation using adaptively scaled radial basis functions, Appl. Soft Comput. J. 88 (2020) 106050. Available: doi: 10.1016/j.asoc.2019.106050..
[10] S.L. Digabel and S.M. Wild, A Taxonomy of Constraints in Simulation-Based Optimization (2015) 1–14. Available: URL: http://arxiv.org/abs/1505.07881..
[11] G. Chen, X. Han, G. Liu, C. Jiang, and Z. Zhao, An efficient multi-objective optimization method for black-box functions using sequential approximate technique, Appl. Soft Comput. J. 12(1) (2012) 14–27. Available: URL:https://doi.org/10.1016/j.asoc.2011.09.011..
[12] J. Martínez-Frutos, D. Herrero-Pérez, Kriging-based infill sampling criterion for constraint handling in multi-objective optimization, J. Global Optim. 64 (1) (2016) 97–115.
[13] E.C. Garrido-Merchán, D. Hernández-Lobato, Predictive Entropy Search for Multi-objective Bayesian Optimization with Constraints, Neurocomputing 361 (2019) 50–68.
[14] M.A. Gelbart, J. Snoek, and R.P. Adams, Bayesian optimization with unknown constraints, in: Proceedings of the Thirtieth Conference on Uncertainty in Artificial Intelligence, ser. UAI'14. Arlington, Virginia, USA: AUAI Press, 2014, pp. 250–259..
[15] S. Ariafar, J. Coll-Font, D. Brooks, J. Dy, ADMMBO: Bayesian Optimization with Unknown Constraints using ADMM, J. Mach. Learn. Res. 20 (10) (2019) 139–148.
[16] H. Garg, A hybrid GSA-GA algorithm for constrained optimization problems, Inf. Sci. 478 (2019) 499–523. doi: 10.1016/j.ins.2018.11.041..
[17] R. Jiao, S. Zeng, C. Li, A feasible-ratio control technique for constrained optimization, Inf. Sci. 502 (2019) 201–217.
[18] C. Audet and J.E. Dennis, Mesh Adaptive Direct Search Algorithms for Constrained Optimization, SIAM J. Optim. 17(1) (2006) 188–217. Available: URL: http://epubs.siam.org/doi/10.1137/040603371..
[19] S. Le Digabel, Algorithm 909: Nomad: Nonlinear optimization with the mads algorithm, ACM Trans. Math. Softw. 37(4) (2011). doi: 10.1145/1916461.1916468..
[20] C. Audet, S.L. Digabel, V.R. Montplaisir, and C. Tribes, NOMAD version 4: Nonlinear optimization with the MADS algorithm, 2021..

[21] C. Audet, A Survey on Direct Search Methods for Blackbox Optimization and Their Applications. New York, NY: Springer, New York, 2014, pp. 31–56. Available: doi: 10.1007/978-1-4939-1124-0_2..
[22] S. Alarie, C. Audet, A.E. Gheribi, M. Kokkolaras, and S. Le Digabel, Two decades of blackbox optimization applications, EURO J. Comput. Optim. 9 (2021) 100011..
[23] Y. Li, Y. Wu, J. Zhao, L. Chen, A Kriging-based constrained global optimization algorithm for expensive black-box functions with infeasible initial points, J. Global Optim. 67 (1–2) (2017) 343–366.
[24] F. Boukouvala and M.G. Ierapetritou, Derivative-free optimization for expensive constrained problems using a novel expected improvement objective function, AIChE J. 60(7) (2014) 2462–2474. Available: URL: https://onlinelibrary.wiley.com/doi/10.1002/aic.14442..
[25] R. Shi, L. Liu, T. Long, Y. Wu, and Y. Tang, Filter-based adaptive Kriging method for black-box optimization problems with expensive objective and constraints, Comput. Methods Appl. Mech. Eng. 34 (2019) 782–805. Available: doi: 10.1016/j.cma.2018.12.026..
[26] H. Dong, P. Wang, C. Fu, and B. Song, Kriging-assisted teaching-learning-based optimization (KTLBO) to solve computationally expensive constrained problems, Inf. Sci. 556 (2021) 404–435. Available: doi: 10.1016/j.ins.2020.09.073..
[27] B. Shahriari, K. Swersky, Z. Wang, R.P. Adams, and N. de Freitas, Taking the Human Out of the Loop: A Review of Bayesian Optimization, Proc. IEEE 104 (1) (2016) 148–175. Available: URL: https://ieeexplore.ieee.org/document/7352306/..
[28] D. Zhan and H. Xing, Expected improvement for expensive optimization: a review, J. Global Optim. 78(3) (2020) 507–544. Available: doi: 10.1007/s10898-020-00923-x..
[29] R.B. Gramacy and H.K.H. Lee, Optimization Under Unknown Constraints, in Bayesian Statistics. Oxford University Press, Oct 2011, vol. 9780199694, no. 1, pp. 229–256. Available: URL: https://oxford.universitypressscholarship.com/view/10.1093/acprof:oso/9780199694587.001.0001/acprof-9780199694587-chapter-8..
[30] J.R. Gardner, M.J. Kusner, Z. Xu, K.Q. Weinberger, and J.P. Cunningham, Bayesian optimization with inequality constraints, in: Proceedings of the 31st International Conference on International Conference on Machine Learning – Volume 32, ser. ICML'14. JMLR.org, 2014, p. II–937–II–945..
[31] P. Jiang, Y. Cheng, J. Yi, and J. Liu, An efficient constrained global optimization algorithm with a clustering-assisted multiobjective infill criterion using Gaussian process regression for expensive problems, Inf. Sci. 569 (2021) 728–745. Available: doi: 10.1016/j.ins.2021.05.015..
[32] C. Antonio, Sequential model based optimization of partially defined functions under unknown constraints, J. Global Optim. 79 (2) (2019) 281–303.
[33] J.M. Hernández-Lobato, M.A. Gelbart, M.W. Hoffman, R.P. Adams, and Z. Ghahramani, Predictive Entropy Search for Bayesian Optimization with Unknown Constraints, in: Proc. 32nd Int. Conf. Mach. Learn., ser. ICML'15. JMLR.org, 2015, p. 1699–1707..
[34] J.M. Hernández-Lobato, M.A. Gelbart, R.P. Adams, M.W. Hoffman, Z. Ghahramani, A General Framework for Constrained Bayesian Optimization Using Information-Based Search, J. Mach. Learn. Res. 17 (1) (2016) 5549–5601.
[35] R.R. Lam and K.E. Willcox, Lookahead bayesian optimization with inequality constraints, in: Proc. 31st Int. Conf. Neural Information Processing Systems, 2017, pp. 1888–1898..
[36] Y. Zhang, X. Zhang, and P.I. Frazier, Two-step Lookahead Bayesian Optimization with Inequality Constraints, in: Proc. 35st Int. Conf. Neural Information Processing Systems, 2021..
[37] R. Alimo, P. Beyhaghi, T.R. Bewley, Delaunay-based derivative-free optimization via global surrogates. Part III: nonconvex constraints, J. Global Optim. 77 (4) (2020) 743–776.
[38] P. Beyhaghi and T. Bewley, Implementation of Cartesian grids to accelerate Delaunay-based derivative-free optimization, J. Global Optim. 69(4) (2017) 927–949. Available: URL: http://link.springer.com/10.1007/s10898-017-0548-3..
[39] R. Alimo, D. Cavaglieri, P. Beyhaghi, and T.R. Bewley, Design of IMEXRK time integration schemes via Delaunay-based derivative-free optimization with nonconvex constraints and grid-based acceleration, J. Global Optim. 79(3) (2021) 567–591. Available: doi: 10.1007/s10898-019-00855-1..
[40] M. Milanese, C. Novara, Set Membership identification of nonlinear systems, Automatica 40 (6) (2004) 957–975.
[41] L. Sabug, F. Ruiz, and L. Fagiano, Trading-off safety, exploration, and exploitation in learning-based optimization: a Set Membership approach, in: 2021 60th IEEE Conference on Decision and Control (CDC), vol. 2021-Decem. IEEE, Dec 2021, pp. 3586–3591..
[42] C. König, M. Turchetta, J. Lygeros, A. Rupenyan, and A. Krause, Safe and Efficient Model-free Adaptive Control via Bayesian Optimization, in: IEEE Int. Conf. Robotics and Automation (ICRA). IEEE, 2021, pp. 9782–9788. Available: doi: 10.1109/ICRA48506.2021.9561349..
[43] L. Fagiano and C. Novara, Learning a Nonlinear Controller From Data: Theory, Computation, and Experimental Results, IEEE Trans. Autom. Control 61(7) (2016) 1854–1868. Available: URL:http://ieeexplore.ieee.org/document/7271025/..
[44] D. Eriksson and M. Poloczek, Scalable Constrained Bayesian Optimization, in 24th Int. Conf. Artificial Intelligence and Statistics, ser. Proceedings of Machine Learning Research, A. Banerjee and K. Fukumizu, Eds., vol. 130. PMLR, 13–15 Apr 2021, pp. 730–738. Available: URL:https://proceedings.mlr.press/v130/eriksson21a.html..
[45] J.J. Liang, T.P. Runarsson, M. Clerc, P.N. Suganthan, C.A.C. Coello, and K. Deb, Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization Problem Definitions and Evaluation Criteria for the CEC 2006 Special Session on Constrained Real-Parameter Optimization, Nanyang Technological University, Tech. Rep., 2006..
[46] F. Borrelli, A. Bemporad, M. Morari, Predictive Control for Linear and Hybrid Systems, Cambridge University Press, 2017.