Les Cahiers du GERAD ISSN: 0711–2440

Benchmarking constrained, multi-objective and surrogate-assisted derivative-free optimization methods

C. Audet, W. Hare, C. Tribes

G–2025–36

May 2025

La publication de ces rapports de recherche est rendue possible grâce au soutien de HEC Montréal, Polytechnique Montréal, Université McGill, Université du Québec à Montréal, ainsi que du Fonds de recherche du Québec – Nature et technologies.

Dépôt légal – Bibliothèque et Archives nationales du Québec, 2025
– Bibliothèque et Archives Canada, 2025

The publication of these research reports is made possible thanks to the support of HEC Montréal, Polytechnique Montréal, McGill University, Université du Québec à Montréal, as well as the Fonds de recherche du Québec – Nature et technologies.

Legal deposit – Bibliothèque et Archives nationales du Québec, 2025
– Library and Archives Canada, 2025

GERAD HEC Montréal
3000, chemin de la Côte-Sainte-Catherine
Montréal (Québec) Canada H3T 2A7

Tél. : 514 340-6053
Téléc. : 514 340-5665
info@gerad.ca
www.gerad.ca

Benchmarking constrained, multi-objective and surrogate-assisted derivative-free optimization methods

Charles Audet [a]

Warren Hare [b]

Christophe Tribes [a]

[a] *GERAD & Département de mathématiques et génie industriel, Polytechnique Montréal, Montréal (Qc), Canada, H3C 3A7*

[b] *Department of Mathematics, University of British Columbia, Okanagan campus, Kelowna (BC), Canada, V1V 1V7*

charles.audet@polymtl.ca
warren.hare@ubc.ca
christophe.tribes@polymtl.ca

May 2025
Les Cahiers du GERAD
G–2025–36

Abstract : Benchmarking is essential for assessing the effectiveness of optimization algorithms. This is especially true in derivative-free optimization, where target problems are often complex simulations that require extensive time to evaluate. This limits the number of evaluations that can be performed, making it critical to have a good understanding of the potential quality of various algorithms. This paper reviews standard benchmarking methods, including convergence plots, performance profiles, data profiles, and accuracy profiles, widely used to evaluate optimization algorithms. The primary contribution of this work is a formal extension of these benchmarking techniques to three specific contexts: constrained optimization, multi-objective optimization, and surrogate-based optimization.

Acknowledgements: This work was partially funded by the NSERC Discover Grants #2020–04448 and #2023–03555, and NSERC/Mitacs Alliance grant #571311–21 in collaboration with Hydro-Québec.

1 Introduction

Benchmarking optimization algorithms is a crucial process in evaluating their performance across various types of problems. Key metrics used for benchmarking combine the quality of the solutions and the effort deployed to produce them. In derivative-free optimization (DFO), the target problems are often the result of a time-consuming simulation known as a blackbox [4]. In consequence, the number of times that the simulation can be evaluated is limited, and the effort deployed by the algorithm is often measured in terms of the number of calls to the simulation.

Existing benchmarking techniques have been developed to provide insights into how algorithms perform relative to these metrics [5, 10, 14, 15, 19]. The present work recalls convergence plots, performance profiles, data profiles, and accuracy profiles for benchmarking optimization algorithms. The main contribution of this work is to formally extend these techniques to three situations: (i) constrained optimization; (ii) multi-objective optimization; (iii) surrogate-assisted optimization.

This document is divided into three main parts. Section 2 discusses how the data necessary for benchmarking needs to be collected and set up. Section 3 reviews existing benchmarking methods in the context of unconstrained or bound-constrained optimization. Section 4 proposes ways to extend these techniques to the three above-mentioned situations. In addition, some concluding remarks appear in Section 5. The data to construct each figure is anonymized, and the open-source code available at `https://github.com/bbopt/RunnerPost`. Data is designed to highlight key attributes that can arise in practice.

2 Benchmarking setup

Consider the case where one wishes to benchmark a finite number of optimization algorithms on problems of the form

$$\min_{x \in X} \{ f(x) \ : \ c(x) \leq 0 \} \tag{1}$$

where $f : X \to \mathbb{R} \cup \{\infty\}$ is the objective function, $c : X \to \mathbb{R}^m \cup \{\infty\}$ is the quantitative constraint function and $X \subseteq \mathbb{R}^n$ is the domain of these functions. The feasible region is denoted by $\Omega = \{x \in X : c(x) \leq 0\}$.

In the present work, an *evaluation* refers to the process that takes a trial point $x \in \mathbb{R}^n$, then computes $f(x)$, $c(x)$, and determines whether x belongs to Ω or not. The present work focuses on the case where the algorithms are DFO algorithms, so this is the only information passed to the algorithm. In real-world applications for such algorithms, it is typically assumed that an evaluation is the most computational expensive portion of the algorithm. As such, a good algorithm is one that achieves a solution (of the desired accuracy) using as few evaluations as possible.

2.1 Test set and algorithms

A *test problem* of the form (1), is defined by its objective function and by its feasible region. A *test problem instance* is defined by pairing a test problem with appropriate conditions, such as initial points or fixing the random seeds used during simulations. A *test set* is a finite collection of test problem instances. For the remainder of this document, $\mathcal{P}$ denotes a test set.

Let $\mathcal{A}$ denote the finite set of algorithms that need to be benchmarked over the test set $\mathcal{P}$. Although each algorithm needs to be applicable to every problem instance, their nature can be either different or similar. One may decide to benchmark widely different algorithms, or can compare methods that differ by varying some specific algorithmic parameters.

Of course, having a large test set is preferable to a smaller one, but having a test set that is representative of the ultimate goal is crucial. For example, if the goal is to determine the best algorithm

to apply to minimize road construction costs, then a small test set consisting of road construction cost problems is preferable to a large test set that is unrelated to road design.

The collection of algorithms to be compared is denoted by $\mathcal{A}$. When selecting which algorithms to compare, it is important to keep both the ultimate goal and test set in mind. For example, if the goal is to determine the best algorithm to apply to minimize a simulation where derivatives are unavailable, then even if the test set allows derivatives, only DFO algorithms should be considered. Moreover, comparing DFO algorithms to algorithms that explicitly use derivatives should be avoided. Finally, the benchmarking process can be sensitive to the number of algorithms. It is recommended to keep the number of algorithms very low.

2.2 Benchmarking data

In order to benchmark the algorithms of $\mathcal{A}$ on the test set $\mathcal{P}$, we assume that all algorithms were run on all test problem instances with a comparable stopping criteria, and that the logs were recorded. For comparison purposes, the logs need to contain information relative to each evaluation. Each entry of the log is associated to a trial point in $\mathbb{R}^n$, and needs to contain at least three elements:

i. the evaluation number;
ii. the objective function value;
iii. an indication of if the trial point belongs to Ω or not.

If all function evaluations are recorded, then the evaluation number can be stored via the data line number. If only function evaluations that improve either the objective function value or feasibility are recorded, then the evaluation number must be stored directly.

The objective function value is usually a real number, but can sometimes be flagged as ∞ or NaN. These occur, for example, when a hidden constraint [8, 17] is triggered and the evaluation failed.

A binary flag can be used to indicate if $x \in X$ or not. If there are no quantitative constraint functions, the feasible region Ω is X and obviously, no data regarding $c(x)$ needs to be stored. Otherwise, whether or not $x \in \Omega$ can be stored by a simple boolean value. Alternately, if g is nontrivial, then the log entry associated to the trial point x would also contain the m values of the vector $c(x)$, or possibly NaN if $x \notin \Omega$.

The log entries may also contain additional elements, such as the numerical values of the trial point $x \in \mathbb{R}^n$, and/or the elapsed time since the start of the algorithm.

Most benchmarking tools presented in the sequel rely on the concept of the *accuracy value*.

Definition 2.1 (Accuracy value)**.** Let x^N be the best feasible point found by an optimization algorithm on a problem instance after N function evaluations. The *accuracy value* of the algorithm at evaluation N is

$$f_{\mathsf{acc}}^N \;=\; \frac{f(x^N) - f^0}{f^* - f^0},$$

where f^0 is the baseline objective function value, and f^* is the best known objective function value.

Both parameters, f^0 and f^*, depend on the specific problem instance but are constant across all algorithms being compared. For algorithms that require a feasible initial point x^0, the baseline value is typically set as $f^0 = f(x^0)$. In cases where algorithms do not require a starting point (such as Latin Hypercube Sampling [24]), the baseline f^0 could be defined as $f^0 = \max\{f(x^{\eta_a}) : a \in \mathcal{A}\}$, where $\mathcal{A}$ is the set of algorithms under comparison, and x^{η_a} is the first trial point visited by algorithm $a \in \mathcal{A}$ that belongs to X. The value of f^* can be chosen in different ways. For example,

a) **Instance-based choice:** The simplest method is to set f^* as the best final value achieved by any algorithm in $\mathcal{A}$ on the instance in question.

b) **Cross-instance choice:** Alternatively, f^* can be defined as the best value obtained across all instances of the same problem (e.g., different starting points or random seeds).

c) **Literature-based choice:** If available, f^* might be selected as the best value known in the literature, even if it was not achieved by any algorithm in $\mathcal{A}$.

By construction, the function value for f^* when applying a) is no smaller than the function value for f^* when applying b), which is no smaller than the function value for f^* when applying c).

The value f^N_{acc} belongs to the interval $[0, 1]$ and represents the relative improvement in quality of the best solution found compared to the starting point after N function evaluations. A value of $f^N_{\text{acc}} = 0$ indicates no improvement beyond f^0, while $f^N_{\text{acc}} = 1$ means the algorithm found the best known solution. If all algorithms fail to improve f^0, and f^* is chosen as in case i. above, this results in division by zero errors. In such cases, the problem instance should likely be removed from the test set and flagged as a difficult instance for these algorithms.

3 Existing benchmarking techniques

This section summarizes some existing benchmarking tools, where it is implicitly assumed that all trial points belong to X with $c(x) \leq 0$.

3.1 Convergence plots

Convergence plots are a well-established tool used to assess the performance of the optimization algorithms on a single test problem instance by visualizing the progression of the objective function $f(x^N)$ over the number of evaluations or time. Two example convergence plots are given in Figure 1.

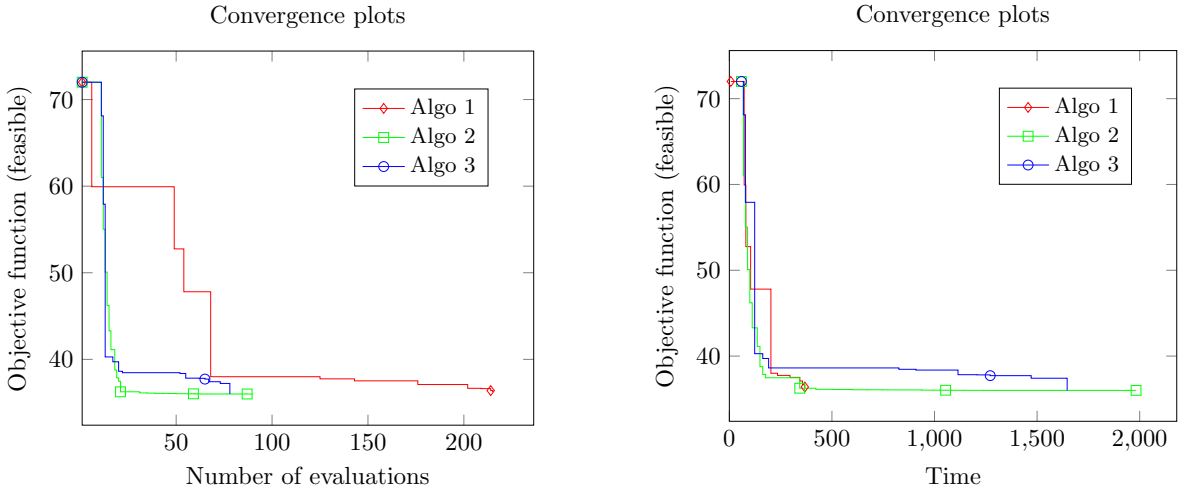

Figure 1: An example of convergence plots

Convergence plots are useful for evaluating both the speed and stability of convergence. They help identify how quickly an algorithm approaches an optimal solution, providing insights into early convergence behavior as well as performance near the optimum. Plotting convergence against elapsed time is particularly useful when an algorithm's computational overhead is significant compared to the time required for function evaluations. For example, Figure 1 suggests that Algo 1 is not competitive when plotting in terms of number of evaluations, but performs very well with respect to time.

These plots, as well as all the ones proposed in this document need to be plotted as monotone staircase functions.

3.2 Accuracy profiles

Accuracy profiles also focus on quality of solution, but instead of working with a single problem instance, aim to summarize quality of solution over the entire test set [5]. These profiles evaluate the final precision attained by the algorithm, making them particularly useful when robustness in achieving high-precision solutions is the main criterion.

Denote the total number of function evaluations used by Algorithm $a \in \mathcal{A}$ on instance $p \in \mathcal{P}$ by $N_{a,p}^{\mathtt{tot}}$. The final solution's accuracy value (Definition 2.1) is then $f_{\mathtt{acc}}^{N_{a,p}^{\mathtt{tot}}}$. The negative of the base 10 logarithm of $1 - f_{\mathtt{acc}}^{N_{a,p}^{\mathtt{tot}}}$ gives an indication of the number of correct decimals of the accuracy value (where $-\log_{10}(0)$ is interpreted as ∞). Indeed, if $d \geq 0$, then

$$-\log_{10}\left(1 - f_{\mathtt{acc}}^{N_{a,p}^{\mathtt{tot}}}\right) \;\geq\; d \qquad \Leftrightarrow \qquad f_{\mathtt{acc}}^{N_{a,p}^{\mathtt{tot}}} \;\geq\; 1 - 10^{-d}.$$

The accuracy profiles are now functions of a variable d, called the *relative accuracy*, that will be compared to the negative of the logarithm.

Definition 3.1 (Accuracy Profile)**.** The accuracy profile of Algorithm $a \in \mathcal{A}$ on the test set $\mathcal{P}$ is defined as the function $r_a : [0, \infty) \to [0, 1]$ via

$$r_a(d) \;=\; \frac{1}{|\mathcal{P}|}\left|\left\{p \in \mathcal{P} \;:\; -\log_{10}\left(1 - f_{\mathtt{acc}}^{N_{a,p}^{\mathtt{tot}}}\right) \geq d\right\}\right|,$$

where $-\log_{10}(0)$ is interpreted as ∞.

To compare the algorithms, the accuracy profiles are plotted on the same graph for all algorithms (see Figure 2).

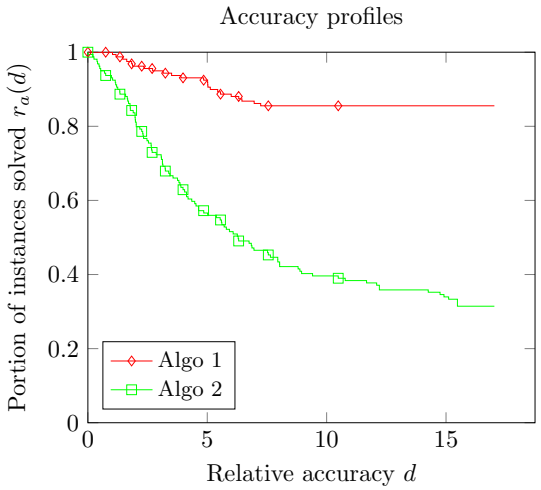

Figure 2: An example of an accuracy profile plot

The x-axis of an accuracy profile plot begins at 0 and should be plotted to no further than machine/software precision (typically $d = 16$). The y-axis of ranges from 0 to 1. All algorithms should begin at $r_a(0) = 1$, as $d = 0$ represents no improvement over starting conditions. Like convergence plots, accuracy profiles are all piecewise constant nonincreasing functions. Curves that are consistently higher correspond to algorithms that consistently produce better results.

In Figure 2, Algo 1 solves 85% of problem instances with 16 digits of accuracy, while Algo 2 only solves 40% of the instances with 10 digits of accuracy.

3.3 Performance profiles

The introduction of performance profiles by Dolan and Moré [10] represents a significant development in the benchmarking of optimization algorithms. Shifting the focus away from accuracy, performance profiles are cumulative graphs that quantify the efficiency (speed of convergence) and robustness (portion of problems solved).

To define performance profiles, the user first selects a *tolerance parameter* $\tau \in [0, 1)$. Using this value, for each pair $a \in \mathcal{A}$ and $p \in \mathcal{P}$, the Boolean variable

$$T_{a,p} \;=\; \begin{cases} 1 & \text{if } f^N_{\mathsf{acc}} \geq 1 - \tau \text{ for some } N, \\ 0 & \text{otherwise} \end{cases}$$

is introduced. Algorithm $a \in \mathcal{A}$ is said to τ-*solve* problem instance $p \in \mathcal{P}$ when $T_{a,p} = 1$. Successfully τ-solving a problem instance implies that the algorithm found a solution that improved the objective function to a point such that $f(x^{N_{a,p}}) \leq f(x^*) + \tau(f(x^0) - f(x^*))$. If $\tau = 0$, then τ-solving a problem instance implies that a solution was found whose objective function value equals f^*.

If Algorithm a τ-solves problem instance p, let $N_{a,p}$ denote the smallest integer such that $f^{N_{a,p}}_{\mathsf{acc}} \geq 1 - \tau$; otherwise set $N_{a,p} = \infty$. Next define

$$r_{a,p} \;=\; \begin{cases} \dfrac{N_{a,p}}{\min\{N_{\tilde{a},p} \,:\, \tilde{a} \in \mathcal{A}, T_{\tilde{a},p} = 1\}} & \text{if } T_{a,p} = 1, \\ \infty, & \text{if } T_{a,p} = 0. \end{cases}$$

Notice that an Algorithm a that τ-solves problem instance p with the least number of function evaluations has the value $r_{a,p} = 1$. In DFO, since the number of function evaluations is integer, it is possible for multiple algorithms to have $r_{a,p} = 1$ on some problems. All other algorithms will have a value of $r_{a,p}$ strictly greater than 1. For example, if $r_{a,p} = 2$, then Algorithm a τ-solved problem instance p using exactly twice as many function evaluations as the algorithm that solved it with the fewest evaluations. Any algorithm that fails to τ-solve the problem will have $r_{a,p} = \infty$.

The performance profile definition can now be stated.

Definition 3.2 (Performance Profile)**.** For a given tolerance $\tau > 0$, the performance profile of Algorithm $a \in \mathcal{A}$ on the test set $\mathcal{P}$ is defined as the function $\rho_a : [1, \infty) \to [0, 1]$ via

$$\rho_a(\alpha) \;=\; \frac{1}{|\mathcal{P}|} \big|\{p \in \mathcal{P} \,:\, r_{a,p} \leq \alpha\}\big|.$$

In other words, $\rho_a(\alpha)$ is the portion of problem instances that Algorithm a τ-solved within a ratio of α function evaluations of the best solver for the respective instance. To compare the algorithms, the performance profiles are plotted on the same graph for all algorithms (see Figure 3). In DFO, typical values for τ are 10^{-1}, 10^{-3}, or 10^{-6}. Ideally, multiple values for τ should be considered.

The x-axis of a performance profile plot begins at $\alpha = 1$ and should extend to the point $\bar{\alpha} > 1$ such that all profiles become constant functions (i.e., $\rho_a(\alpha) = \rho_a(\bar{\alpha})$ for all $\alpha \geq \bar{\alpha}$, $a \in \mathcal{A}$). The y-axis ranges from 0 to 1. Like accuracy profiles, curves that are consistently higher correspond to algorithms that consistently produce better results. In contrast to accuracy profiles, performance profiles are piecewise constant *nondecreasing* functions. The maximum value of a performance profile represents the portion of problems that the algorithm τ-solved.

In Figure 3, the Algo 1 curve is always above that of Algo 2, so would likely be deemed the most successful method. For $\tau = 1\%$, Algo 1 is the fastest to τ-solves 82% of the problem instances. Algo 2 is the fastest on 20% of the instances. It follows that both algorithms τ-solve 2% of the instances using the same number of evaluations (since $82\% + 20\%$ exceeds one by 2%). The plot on the right suggests that the ratio α becomes quite high before the profile for Algo 2 stabilizes.

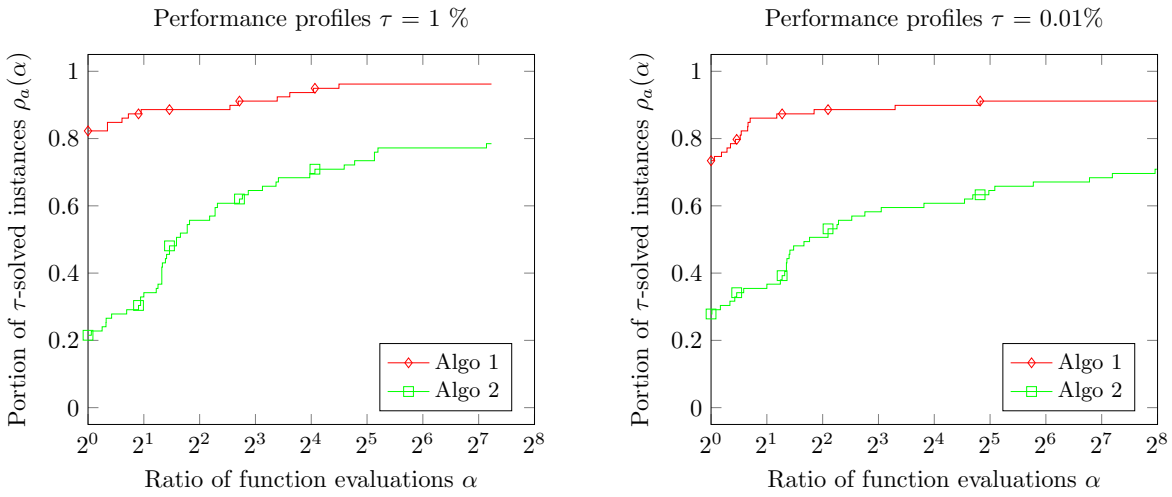

Figure 3: An example of performance profile plots

Since performance profiles visualize algorithms' effectiveness relative to the other algorithms, they are subject to the switching phenomenon [13]. In particular, adding or removing an algorithm from $\mathcal{A}$ can change the relative ranking of other algorithms. The switching phenomenon does not appear in data profiles, presented next.

3.4 Data profiles

Building on the concept of performance profiles, Moré and Wild [19] introduced data profiles, which extend the benchmarking framework by focusing on the computational effort required to achieve a given precision. While performance profiles emphasize efficiency relative to other algorithms, data profiles capture the trade-off between computational effort (such as the number of function evaluations) and the accuracy of the solution.

The information used to create data profiles is similar to the information for performance profiles, but (for DFO) also requires the dimension of problem instance $p \in P$, denoted by n_p.[1]

Definition 3.3 (Data Profile)**.** For a given tolerance $\tau > 0$, the data profile of Algorithm $a \in \mathcal{A}$ on the test set $\mathcal{P}$ is defined as the function $d_a : \mathbb{N} \to [0, 1]$ via

$$d_a(k) \;=\; \frac{1}{|\mathcal{P}|} \left| \{p \in \mathcal{P} \,:\, N_{a,p} \leq k\,(n_p + 1)\,T_{a,p}\} \right|.$$

The value $d_a(k)$ is the portion of problem instances that Algorithm $a \in \mathcal{A}$ τ-solved within k groups of $n_p + 1$ function evaluations. To compare the algorithms, the data profiles are plotted on the same graph for all algorithms, as shown in Figure 4. Similar to performance profiles, multiple values for τ should be considered.

The x-axis of a data profile plot begins at $k = 0$, where $d_a(0) = 0$ for all algorithms $a \in \mathcal{A}$. The x-axis should be extended to the point where all profiles become constant functions. The y-axis ranges from 0 to 1. Like performance profiles, data profiles are piecewise constant nondecreasing functions and profiles that are consistently higher correspond to algorithms that consistently produce better results. And, the maximum value of a data profile represents the portion of problems that the algorithm τ-solved.

[1] The scaling by $(n_p + 1)$ in the definition of d_a is based on the DFO concept of a gradient approximation requiring $(n_p + 1)$ function evaluations [9, 19, 21], making a natural link between iteration complexity and dimension.

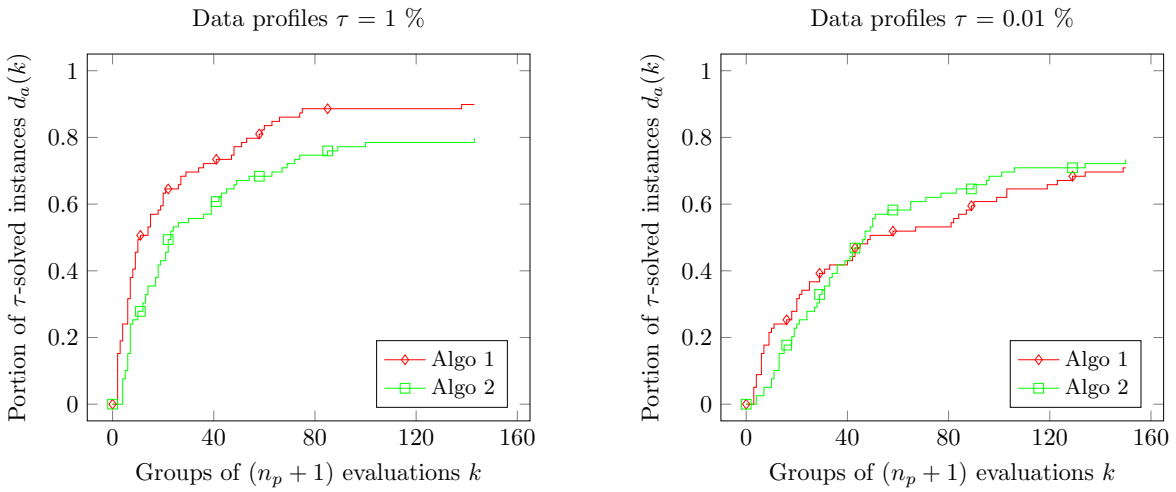

Figure 4: An example of data profile plots

In the left plot of Figure 4, the ALGO 1 curve is above that of ALGO 2. However, in the right plot with a smaller value of τ, ALGO 2 outperforms ALGO 1 when k exceeds 50. This illustrates the importance of visualizing multiple values for τ.

4 Extensions

As mentioned, existing benchmarking techniques implicitly assume that there is no special effort required to acquire feasible points. In addition, the existing techniques focus on single-objective optimization and do not consider the possibility of surrogate functions that could be used in the optimization algorithms. Section 4.1 proposes ways to compare such problems when the initial point is infeasible. Section 4.2 proposes ways to benchmark DFO algorithms that are intended for multi-objective problems. Section 4.3 discusses ways to benchmark DFO algorithms that take advantage of a surrogate optimization problem to generate and order trial points.

4.1 Benchmarking constrained optimization algorithms

This section shows one way to adapt the above benchmarking algorithms for constrained optimization problems that allow for infeasible initial points.

On a given problem instance, define η_a to be the number of blackbox evaluations required by algorithm $a \in \mathcal{A}$ to generate a feasible solution. Observe that $\eta_a = 0$ when the starting point x^0 is feasible, and $\eta_a = \infty$ when the algorithm fails to produce a feasible solution.

Convergence plots can be generalized by making use of the *constraint violation function*. The constraint violation function, $h : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$, was introduced in the context of filter algorithms [11, 12], and is defined as

$$h(x) \;=\; \begin{cases} \displaystyle\sum_{j \in J} \left(\max\{g_j(x), 0\}\right)^2 & \text{if } \; x \in \Omega, \\ \infty & \text{otherwise,} \end{cases}$$

where g and Ω are defined in Problem (1), and $J = \{1, 2, \ldots, m\}$ are the indices of the quantifiable constraints [17].

To adjust convergence plots to this style of algorithm, on the left part of the figure, plot the constraint violation function value h in terms of the number of evaluations, from 0 to η_a, and on the right side, plot the best feasible objective function value f in terms of the number of evaluations

starting at η_a. An example appears in Figure 5. ALGO 1 is the fastest to produce a feasible solution, and once that it did, it rapidly finds a feasible solution with an objective function value close to 4.4. ALGO 2 uses approximately 250 evaluations to reach feasibility, and struggles to find a good solution.

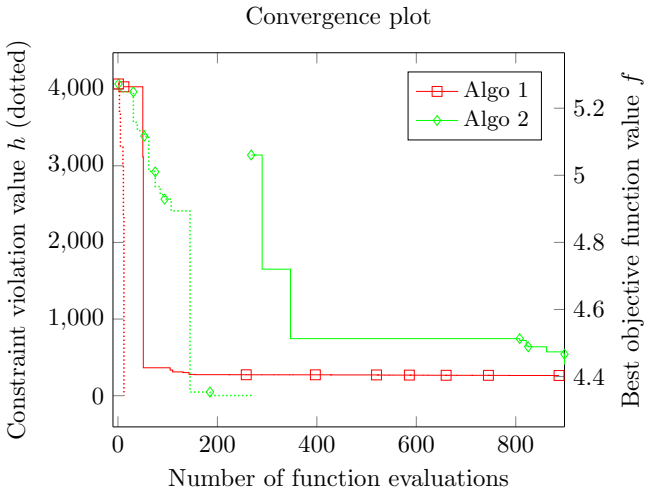

Figure 5: An example of a convergence plot for algorithms that allow infeasible initial points

The other profiles rely on the accuracy value, which depends on f^0 and f^*. There are no issues with selecting f^* as methods (a), (b), and (c), on page 3, are still sound. But in the situation where x^0 is infeasible, it does not make sense to use $f^0 = f(x^0)$ because that value might not be an upper bound for f^*. Different approaches for defining f^0 can be used, with the most important aspect being that all algorithms and test problem instances are treated equally. One approach for benchmarking constrained optimization algorithms would be define it as

$$f^0 \;=\; \min\{f(x^{\eta_a}) \,:\, a \in \mathcal{A}, \eta_a < \infty\}.$$

Another approach defines it as

$$f^0 \;=\; \max\{f(x^{\eta_a}) \,:\, a \in \mathcal{A}, \eta_a < \infty\}.$$

Using the max has the advantage of ensuring that all accuracy function values f^N_{acc} are in $[0, 1]$, but has the drawback of systematically generating accuracy values close to 1 when the first feasible point generated by an algorithm has a very high objective function value.

Figure 6 illustrates two accuracy profile plots, constructed using the same data, but using these two different methods for selecting f^0. Observe that both profiles suggest the same conclusion: (ALGO 1 is preferable to ALGO 2). The choice of f^0 only causes minor changes to the profiles.

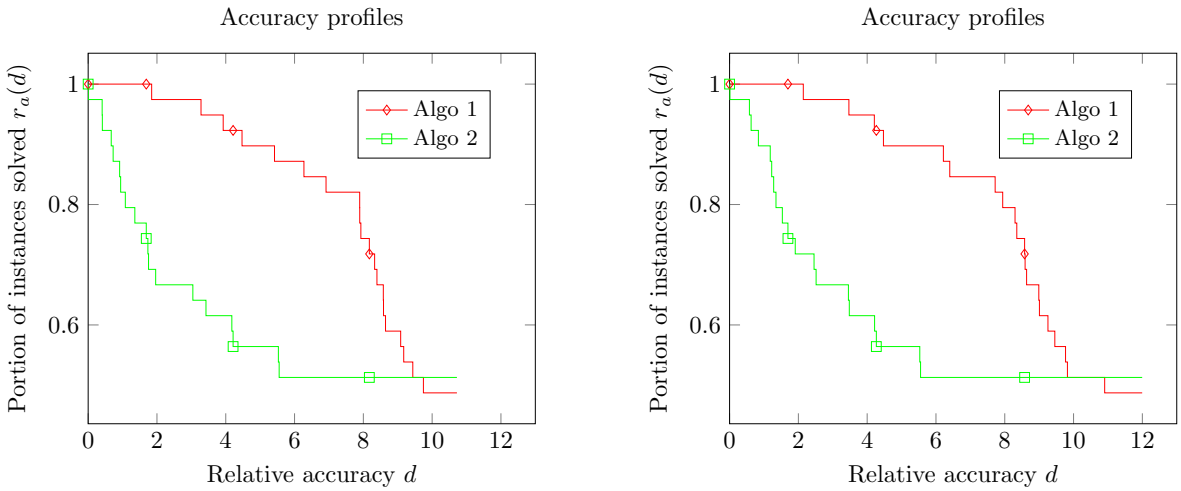

Figure 6: Two examples of accuracy profile plots for algorithms that allow infeasible initial points: $f^0 = \min\{f(x^{\eta_a}) : a \in \mathcal{A}, \eta_a < \infty\}$ **(left);** $f^0 = \max\{f(x^{\eta_a}) : a \in \mathcal{A}, \eta_a < \infty\}$ **(right)**

4.2 Benchmarking multi-objective optimization algorithms

In certain modeling situations, it is desirable to optimize more than one objective. This creates a multi-objective optimization problem [18] of the form

$$\min_{x \in \Omega} \quad F(x) \tag{2}$$

where $\Omega \subseteq \mathbb{R}^n$ is the feasible region and $F(x) = \left(f^{(1)}(x), \ldots, f^{(\eta)}(x)\right)$ is composed of the $\eta \geq 2$ objective functions $f^{(i)} : \mathbb{R}^n \to \mathbb{R} \cup \{\infty\}$ for $i \in \{1, 2, \ldots, \eta\}$.

Typically, there does not exist a single vector $x \in \Omega$ that simultaneously optimizes all objectives. Therefore, the solution to the multi-objective problem is composed of a set of tradeoff points, chosen based on the Pareto dominance criterion [25], which is used to compare two decision vectors $u \in \Omega$ and $v \in \Omega$. A vector $u \in \Omega$ is said to dominate another vector $v \in \Omega$ (denoted $u \succ v$) if the values of all objective functions evaluated at u are less than or equal to those at v, with at least one of the inequalities being strict.

The set of non-dominated points forms the solution set of the multi-objective problem, known as the *Pareto set* denoted by $\Omega_{\mathcal{P}}$. Its image in $\mathbb{R}^\eta$ under the mapping F is called the *Pareto front* and is denoted by $F_{\mathcal{P}} = F(\Omega_{\mathcal{P}}) \subseteq \mathbb{R}^\eta$.

There are many indicators to quantify the quality of Pareto front approximations [3, 20]. One of the most popular is the *hypervolume indicator* (called the *area measure* when two objectives are considered, and *volume indicator* when three objectives are considered), and requires the following definitions. Let $\tilde{\Omega}_{\mathcal{A}}$ be the set of non-dominated points of $\cup_{a \in A} \tilde{\Omega}_a$, where $\tilde{\Omega}_a$ is the Pareto set approximation produced by algorithm $a \in A$. The *ideal* and *nadir* points are the vectors in $\mathbb{R}^\eta$ given by

$$\begin{aligned} \mathcal{I} &= \left(\mathcal{I}^{(1)}, \ldots, \mathcal{I}^{(\eta)}\right) = \left(\min_{x \in \tilde{\Omega}_{\mathcal{A}}} f^{(1)}(x), \ldots, \min_{x \in \tilde{\Omega}_{\mathcal{A}}} f^{(\eta)}(x)\right) \in \mathbb{R}^\eta \\ \text{and} \quad \mathcal{N} &= \left(\mathcal{N}^{(1)}, \ldots, \mathcal{N}^{(\eta)}\right) = \left(\max_{x \in \tilde{\Omega}_{\mathcal{A}}} f^{(1)}(x), \ldots, \max_{x \in \tilde{\Omega}_{\mathcal{A}}} f^{(\eta)}(x)\right) \in \mathbb{R}^\eta. \end{aligned}$$

The hyperrectangle $R = [\mathcal{I}, \mathcal{N}] \subset \mathbb{R}^\eta$ contains the Pareto front approximation $F(\tilde{\Omega}_{\mathcal{A}})$.

For a given Pareto front approximation, the hypervolume indicator computes the hypervolume of the dominated zone within the rectangle R, and divides it by the area of R. An algorithm whose score is the larger would be considered to have provided the better Pareto front approximation.

Definition 4.1. For algorithm $a \in \mathcal{A}$, ideal point $\mathcal{I}$, and nadir point $\mathcal{N}$, the *hypervolume indicator* of the Pareto front approximation $\widetilde{F}^a_{\mathcal{P}}$ is

$$s_a \;=\; \frac{1}{V} \iint_R \iota_F(u) du$$

where

$$R = [\mathcal{I}, \mathcal{N}], \qquad V = \mathrm{vol}(R) = \prod_{i=1}^{N} \left(\mathcal{N}^{(i)} - \mathcal{I}^{(i)} \right)$$

and

$$\iota_F(u) = \begin{cases} 1 & \text{there exists } v \in \widetilde{F}^a_{\mathcal{P}} \text{ such that } u \succ v \\ 0 & \text{otherwise.} \end{cases}$$

In Definition 4.1, V is the hypervolume of R and the multivariate integral $\iint_R \iota_F(u) du$ computes the hypervolume of the dominated region within R. Thus s_a is the portion of R that is dominated by the Pareto front approximation.

In the case where exactly two objectives are considered, and the Pareto set approximation $\Omega^a_{\mathcal{P}}$ is finite, let $\{x^1_a, \ldots, x^m_a\}$ be the subset of $\Omega^a_{\mathcal{P}}$ whose image by F is included in R, ordered by increasing values of $f^{(1)}$. The integral in Definition 4.1 can be computed via

$$\iint_\Omega \iota_F(u) du = \sum_{i=1}^{m} s^i_a \;\text{ with }\; s^i_a \;=\; \begin{cases} \left(f^{(1)}(x^{i+1}_a) - f^{(1)}(x^i_a)\right)\left(\mathcal{N}^{(2)} - f^{(2)}(x^i_a)\right) & \text{if } \; i < m \\ \left(\mathcal{N}^{(1)} - f^{(1)}(x^m_a)\right)\left(\mathcal{N}^{(2)} - f^{(2)}(x^m_a)\right) & \text{if } \; i = m. \end{cases}$$

If three or more objectives are considered, or the Pareto front is infinite, the hypervolume indicator can be quite difficult to compute [22].

For benchmarking purposes, the ideal and nadir points used to construct the area measure for a specific test instance have a fundamental impact on the computed hypervolume indicator. As such, it is critical to determine these points in a fair and consistent manner. If a theoretical true solution exists, then the ideal and nadir points can be extracted from that solution. Otherwise, the ideal and nadir points can be approximated using the union of all Pareto front approximations over all algorithms,

$$\widetilde{F}_{\mathcal{P}} \;=\; \bigcup_{a \in \mathcal{A}} \widetilde{F}^a_{\mathcal{P}}.$$

In order to benchmark multi-objective algorithms, one can redefine the accuracy function from Definition 2.1. First, for each problem instance, select the representative ideal and nadir points $\mathcal{I}$ and $\mathcal{N}$. Let s^* be the hypervolume indicator of the Pareto front of $\widetilde{F}_{\mathcal{P}}$ and s^0 be the hypervolume indicator of the *initial* Pareto front approximation. The optimal Pareto used to compute s^* should be the same as that used to compute the ideal and nadir points. The Pareto front used to compute s^0 could consist of a single point, or a collection of starting points generated through some initial sampling. In any case, all algorithms should be given the same initial point(s), so s^0 should be well-defined. It is possible that $s^0 = 0$, in the event that the images of the initial points are outside of the rectangle R.

Let $s_a(x^N)$ be the hypervolume indicator of the Pareto Front approximation corresponding to the N^{th} function evaluation of algorithm $a \in \mathcal{A}$. The accuracy function from Definition 2.1 can now be redefined as

$$s^N_{\mathtt{acc}} \;=\; \frac{s_a(x^N) - s^0}{s^* - s^0}.$$

Figure 7 presents two example performance profiles plots using this technique, with the same value of τ. The plot on the left compares three algorithms. For α close to one, Algo 1 dominates the two other algorithms, as it τ-solved more than 50% of the instances the fastest. The performances of the two other algorithms is difficult to distinguish. The plot on the right only compares Algo 2 and

ALGO 3. Now that ALGO 1 is absent, the figure reveals that ALGO 2 clearly dominates ALGO 3 for low values of α, as it was the fastest to τ-solve two thirds of the instances. For large values of α, the relative performance of the algorithms is not significantly different in both figures. However, notice the range of the ratio in both figures: 40 versus 12. The difference is due to the fact that ALGO 1 is very quick to τ-solve instances.

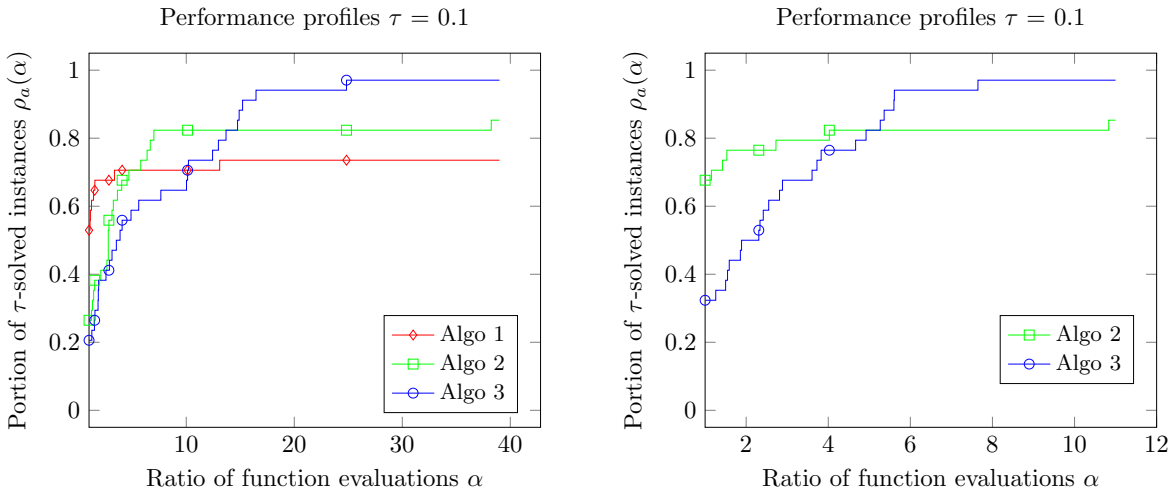

Figure 7: Two examples of performance profile plots for multi-objective optimization algorithms:(left) three algorithms ; (right) two algorithms

4.3 Benchmarking surrogate-assisted optimization algorithms

Surrogate-based optimization assumes the existence of *surrogate functions*; i.e., cheaper-to-evaluate functions that can be used as approximations for the true objective and/or constraint functions [6, 23, 16]. These surrogates can be static simplifications or dynamic models built through regression, interpolation, Krieging or Bayesian approximations based on previously evaluated points. Some methods also incorporate feasibility prediction models and adaptive strategies to improve accuracy and robustness in constrained optimization problems. By evaluating these surrogates, promising candidate points for the true Problem (1) can be identified. This section shows how to adapt benchmarking tools to account for the use of surrogates.

Convergence plots and accuracy profiles should be constructed using only the evaluations of the true problem since they do not consider the effort deployed by the algorithm. However, performance and data profiles represent the proportion of τ-solved problem instances in terms of a measure of the effort deployed.

Dynamic surrogates, such as Gaussian or quadratic models, are generally much quicker to evaluate than the true blackbox problems. However, the cost of repeatedly building these dynamic surrogates can be time-consuming. One way to adapt the profiles is to plot the proportion of τ-solved problem instances in terms of computational time instead of in terms of groups of $n_p + 1$ evaluations. For performance profiles, this is the relative computational time compared to the champion method, while for data profiles, it would be in terms of elapsed time. The latter naturally depends on the instance sizes and is more appropriate when comparing instances of the same problem (using various starting points and/or random seeds).

Adjusting performance and data profiles to account for the use of time-consuming static surrogate problems involves redefining how function calls are counted. This can be formalized as follows:

Let us pause the deployment of an algorithm on a problem instance, and define

- N_t as the current number of evaluations of the true blackbox;
- N_s as the current number of evaluations of the surrogate;
- $w \in [0, 1)$ as a user-defined weight factor reflecting the relative cost of surrogate evaluations compared to true evaluations.

The effective number of evaluations, N, is then defined as a weighted sum of the true and surrogate evaluations:

$$N \;=\; N_t + w\, N_s.$$

In this formula, w should be selected to accurately represent the ratio of effort it takes to compute the surrogate function(s) versus the effort it takes to compute the true function(s). By using the value N to compute the accuracy value in Definition 2.1 (instead of N_t), profiles are created that more accurately reflect the computational effort by considering both types of evaluations; thereby accounting for the surrogate's contributions to the optimization process while maintaining a consistent framework for comparing different optimization strategies.

Figure 8 presents two examples of data profiles using this technique. The left one shows a situation in which the surrogate is instantaneous to evaluate ($w = 0$), while the right shows the effect of setting $w = \frac{1}{10}$ (representing that the surrogate function is 10 times cheaper to evaluate than the true function). Both plots are generated using the same evaluation budget. As a consequence, the profiles with $w = \frac{1}{10}$ exhaust the evaluation budget and produce inferior quality solutions. The profile for ALGO 1 increases more rapidly when the cost of the surrogate is null. Careful inspection of the figure reveals that, the curve for ALGO 1 reaches its maximal value of 0.56 at approximately 800 groups of $n_p + 1$ evaluations when $w = 0$, and only attains 0.52 near 1100 groups of $n + 1$ evaluations when $w = \frac{1}{10}$.

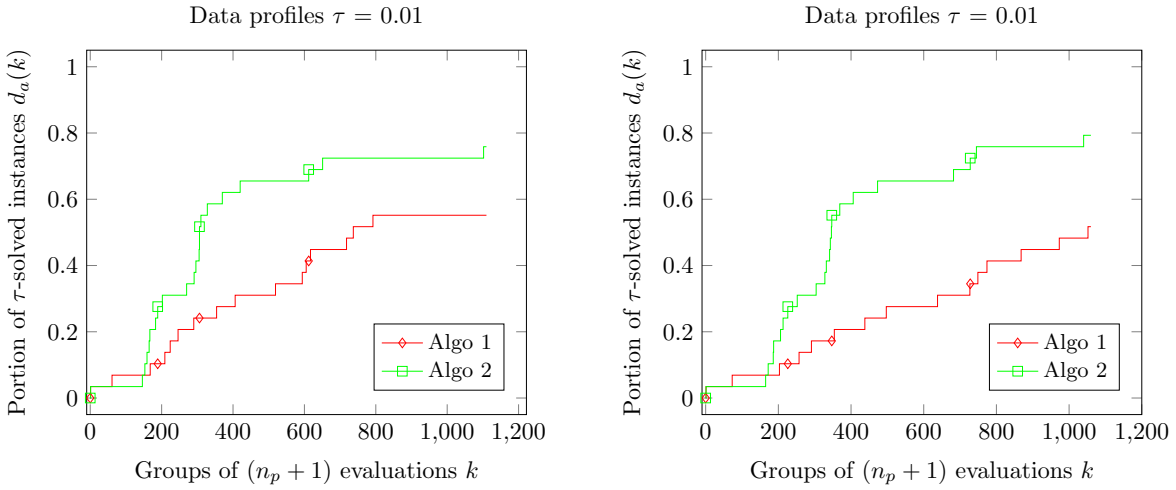

Figure 8: Two examples of data profile plots for surrogate-based optimization algorithms:(left) $w = 0$; **(right)** $w = \frac{1}{10}$

There are situations in which the above strategy to compensate the cost of a surrogate is not appropriate. For example, situations arise when there is not one, but there are a collection of surrogate problems. In this case, the approach is easily extended by applying additional weight parameters. If the surrogate involves controlling how many repetitions are used within some form of Monte Carlo simulation (as seen in [1, 2, 7]), then N can be set as the number of repetitions.

5 Discussion

This paper provides a comprehensive review and extension of benchmarking techniques for DFO algorithms, focusing on three specific contexts: constrained optimization, multi-objective optimization, and surrogate-assisted optimization. Established tools such as convergence plots, accuracy profiles, performance profiles, and data profiles, were adapted to address the challenges present in these specific contexts.

For constrained optimization, we introduced adaptations that handle infeasible initial points and hidden constraints, emphasizing consistency in defining baseline and best-known solutions. For multi-objective optimization, we use a measure to quantify the quality of Pareto front approximations. For surrogate-assisted optimization, we highlighted the importance of accounting for all evaluation costs.

The methods presented in this paper aim to standardize and elevate the benchmarking process, ensuring that algorithmic comparisons are fair, insightful, and aligned with practical optimization needs. We comment that the techniques can be easily combined to deal with situations like multi-objective surrogate-based optimization. We hope that these contributions will serve as a foundation for continued advancements in optimization algorithm research.

Appendix: Software availability and usage

In order to facilitate the use of the techniques herein, software can be found at `https://github.com/bbopt/RunnerPost`.

The RunnerPost software repository primarily consists of C++ code, supplemented by Python. The code is designed for high-performance computational tasks, utilizing C++ for core processing and an optional Python interface for scripting and automation. CMake is used for building the project, and Cython facilitates the integration between C++ and Python. The code supports both Python and command-line interfaces. The code performs post-processing and profiling of optimization results provided as text files and returns post-processed text files that support LaTeX for producing PDF files. All profile plots in this paper have been generated by RunnerPost.

Documentation on installation and utilization of the code are described in the repository's README file. The project is licensed under the MIT License. We welcome suggestions and contributions that improve the project.

References

[1] S. Alarie, C. Audet, P.-Y. Bouchet, and S. Le Digabel. Optimisation of stochastic blackboxes with adaptive precision. SIAM Journal on Optimization, 31(4):3127–3156, 2021.

[2] N. Andrés-Thió, C. Audet, M. Diago, A.E. Gheribi, S. Le Digabel, X. Lebeuf, M. Lemyre Garneau, and C. Tribes. `solar`: A solar thermal power plant simulator for blackbox optimization benchmarking. Technical Report G-2024-37, Les cahiers du GERAD, 2025. To appear in Optimization and Engineering.

[3] C. Audet, J. Bigeon, D. Cartier, S. Le Digabel, and L. Salomon. Performance indicators in multiobjective optimization. European Journal of Operational Research, 292(2):397–422, 2021. Invited Review.

[4] C. Audet and W. Hare. Derivative-Free and Blackbox Optimization. Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2017.

[5] V. Beiranvand, W. Hare, and Y. Lucet. Best practices for comparing optimization algorithms. Optimization and Engineering, 18(4):815–848, 2017.

[6] A.J. Booker, J.E. Dennis, Jr., P.D. Frank, D.B. Serafini, V. Torczon, and M.W. Trosset. A Rigorous Framework for Optimization of Expensive Functions by Surrogates. Structural and Multidisciplinary Optimization, 17(1):1–13, 1999.

[7] X. Chen and C.T. Kelley. Optimization with hidden constraints and embedded Monte Carlo computations. Optimization and Engineering, 17(1):157–175, 2016.

[8] T.D. Choi and C.T. Kelley. Superlinear convergence and implicit filtering. SIAM Journal on Optimization, 10(4):1149–1162, 2000.

[9] A.L. Custódio, J.E. Dennis, Jr., and L.N. Vicente. Using simplex gradients of nonsmooth functions in direct search methods. IMA Journal of Numerical Analysis, 28(4):770–784, 2008.

[10] E.D. Dolan and J.J. Moré. Benchmarking optimization software with performance profiles. Mathematical Programming, 91(2):201–213, 2002.

[11] R. Fletcher and S. Leyffer. Nonlinear programming without a penalty function. Mathematical Programming, Series A, 91:239–269, 2002.

[12] R. Fletcher, S. Leyffer, and Ph.L. Toint. A brief history of filter methods. SIAM SIAG/OPT Views-and-News, 18(1):2–12, 2006.

[13] N. Gould and J. Scott. A note on performance profiles for benchmarking software. ACM Transactions on Mathematical Software, 43(2):15:1–15:5, 2016.

[14] N. Hansen. The CMA Evolution Strategy: A Comparing Review. In J. Lozano, P. Larrañaga, I. Inza, and E. Bengoetxea, editors, Towards a New Evolutionary Computation, volume 192 of Studies in Fuzziness and Soft Computing, pages 75–102. Springer, Berlin, Heidelberg, 2006.

[15] N. Hansen, A. Auger, R. Ros, O. Mersmann, T. Tušar, and D. Brockhoff. COCO: a platform for comparing continuous optimizers in a black-box setting. Optimization Methods and Software, 36(1):114–144, 2021.

[16] J. Larson, M. Menickelly, and S.M. Wild. Derivative-free optimization methods. Acta Numerica, 28:287–404, 2019.

[17] S. Le Digabel and S.M. Wild. A taxonomy of constraints in black-box simulation-based optimization. Optimization and Engineering, 25(2):1125–1143, 2024.

[18] K. Miettinen. Nonlinear Multiobjective Optimization. Springer, 1999.

[19] J.J. Moré and S.M. Wild. Benchmarking Derivative-Free Optimization Algorithms. SIAM Journal on Optimization, 20(1):172–191, 2009.

[20] T. Okabe, Y. Jin, and B. Sendhoff. A critical survey of performance indices for multi-objective optimisation. In Evolutionary Computation, volume 2, pages 878–885, Canberra, Australia, 2003.

[21] R.G. Regis. The calculus of simplex gradients. Optimization Letters, 9(5):845–865, 2015.

[22] P.K. Shukla, N. Doll, and H. Schmeck. A Theoretical Analysis of Volume Based Pareto Front Approximations. In Proceedings of the 2014 Annual Conference on Genetic and Evolutionary Computation, pages 1415–1422, New York, NY, USA, 2014. ACM.

[23] A. Sóbester, A.I.J. Forrester, D.J.J. Toal, E. Tresidder, and S. Tucker. Engineering design applications of surrogate-assisted optimization techniques. Optimization and Engineering, 15(1):243–265, 2014.

[24] B. Tang. Orthogonal array-based latin hypercubes. Journal of the American Statistical Association, 88(424):1392–1397, 1993.

[25] P.L. Yu. Cone convexity, cone extreme points and nondominated solutions in decision problems with multi-objectives. Journal of Optimization Theory and Application, 14:319–377, 1974.