

Benchmarking bilevel derivative-free optimization algorithms

C. Audet, V. Dijon, Y. Diouane

G–2026–26

May 2026

La collection *Les Cahiers du GERAD* est constituée des travaux de recherche menés par nos membres. La plupart de ces documents de travail a été soumis à des revues avec comité de révision. Lorsqu'un document est accepté et publié, le pdf original est retiré si c'est nécessaire et un lien vers l'article publié est ajouté.

Citation suggérée : C. Audet, V. Dijon, Y. Diouane (Mai 2026). Benchmarking bilevel derivative-free optimization algorithms, Rapport technique, Les Cahiers du GERAD G– 2026–26, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (<https://www.gerad.ca/fr/papers/G-2026-26>) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

The series *Les Cahiers du GERAD* consists of working papers carried out by our members. Most of these pre-prints have been submitted to peer-reviewed journals. When accepted and published, if necessary, the original pdf is removed and a link to the published article is added.

Suggested citation: C. Audet, V. Dijon, Y. Diouane (May 2026). Benchmarking bilevel derivative-free optimization algorithms, Technical report, Les Cahiers du GERAD G–2026–26, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (<https://www.gerad.ca/en/papers/G-2026-26>) to update your reference data, if it has been published in a scientific journal.

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, 2026
– Bibliothèque et Archives Canada, 2026

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, 2026
– Library and Archives Canada, 2026

Benchmarking bilevel derivative-free optimization algorithms

Charles Audet ^{a, b}

Valentin Dijon ^{a, b}

Youssef Diouane ^{a, b}

^a GERAD, Montréal (Qc), Canada, H3T 1J4

^b Département de mathématiques et de génie industriel, Polytechnique Montréal, Montréal (Qc), Canada, H3T 1J4

charles.audet@polymtl.ca
valentin.dijon@etud.polymtl.ca
youssef.diouane@polymtl.ca

May 2026
Les Cahiers du GERAD
G–2026–26

Copyright © 2026 Audet, Dijon, Diouane

Les textes publiés dans la série des rapports de recherche *Les Cahiers du GERAD* n'engagent que la responsabilité de leurs auteurs. Les auteurs conservent leur droit d'auteur et leurs droits moraux sur leurs publications et les utilisateurs s'engagent à reconnaître et respecter les exigences légales associées à ces droits. Ainsi, les utilisateurs:

- Peuvent télécharger et imprimer une copie de toute publication du portail public aux fins d'étude ou de recherche privée;
- Ne peuvent pas distribuer le matériel ou l'utiliser pour une activité à but lucratif ou pour un gain commercial;
- Peuvent distribuer gratuitement l'URL identifiant la publication.

Si vous pensez que ce document enfreint le droit d'auteur, contactez-nous en fournissant des détails. Nous supprimerons immédiatement l'accès au travail et enquêterons sur votre demande.

The authors are exclusively responsible for the content of their research papers published in the series *Les Cahiers du GERAD*. Copyright and moral rights for the publications are retained by the authors and the users must commit themselves to recognize and abide the legal requirements associated with these rights. Thus, users:

- May download and print one copy of any publication from the public portal for the purpose of private study or research;
- May not further distribute the material or use it for any profit-making activity or commercial gain;
- May freely distribute the URL identifying the publication.

If you believe that this document breaches copyright please contact us providing details, and we will remove access to the work immediately and investigate your claim.

Abstract : Bilevel optimization involves an upper-level and a lower-level decision maker. The lower-level optimization problem is nested within the constraints of the upper-level one. A point is said to be admissible for the bilevel problem if it satisfies all constraints and is optimal for the lower-level decision-maker. Bilevel derivative-free optimization (BL-DFO) algorithms address bilevel optimization problems in which either the upper-level or the lower-level problem is solved using a derivative-free optimization method. In this context, existing BL-DFO benchmarking techniques often do not rigorously validate the admissibility of proposed solutions, and do not adequately account for the computational effort deployed by the upper- and lower-level solvers. This work proposes a benchmarking methodology for BL-DFO algorithms. A post-optimization procedure, named refereeing procedure, is introduced to discard non-admissible points and ensure a fair comparison between the algorithms. The computational effort deployed by upper- and lower-level solvers are also taken into account into the overall computational cost. Numerical experiments illustrate the benchmarking methodology.

Keywords : Bilevel optimization; derivative-free optimization; benchmarking; performance evaluation; computational budget

Résumé : L'optimisation biniveau implique un décideur de haut-niveau et un décideur bas-niveau. Le problème d'optimisation bas-niveau est compris dans les contraintes du problème haut-niveau. Un point est dit admissible pour le problème biniveau s'il satisfait toutes les contraintes et s'il est optimal pour le décideur bas-niveau. Les algorithmes d'optimisation biniveau sans dérivées (BL-DFO) résolvent les problèmes d'optimisation biniveau où soit le problème haut-niveau, soit le problème bas-niveau, est résolu à l'aide d'une méthode d'optimisation sans dérivées. Dans ce contexte, les techniques existantes de benchmarking des algorithmes BL-DFO ne vérifient pas de manière rigoureuse l'admissibilité des solutions proposées. De plus, elles ne tiennent pas compte correctement de l'effort déployé par les solveurs aux deux niveaux. Ce travail propose une méthodologie de benchmarking pour les algorithmes BL-DFO. Une procédure post-optimisation, appelée procédure d'arbitrage, est introduite pour révoquer les points non admissibles et garantir une comparaison équitable entre les algorithmes. L'effort déployé par les solveurs de niveau supérieur et inférieur est également pris en compte dans le coût de calcul global. Des expériences numériques illustrent la méthodologie de benchmarking.

Mots clés : Optimisation biniveau ; optimisation sans dérivées ; benchmarking ; évaluation de performances ; budget computationnel

Acknowledgements: The authors declare that they have no conflicts of interest or competing interests. This research is supported by the NSERC discovery grants RGPIN-2026-05340 (Audet) and RGPIN-2024-05093 (Diouane).

1 Introduction

Bilevel optimization (BLO) is a hierarchical optimization framework in which a subset of the decision variables is controlled by a second decision-maker, guided by its own objective function and subject to its own constraints. BLO has been extensively studied in the literature (e.g. [14, 19, 32, 39]) and arises in various application domains. BLO naturally models the leader-follower interactions from game theory, especially Stackelberg games [47, 53, 56]. Other applications in engineering also arise from this structure when the two problems are in conflict [2, 41, 60]. In recent years, BLO has gained renewed attention due to its relevance in machine learning and artificial intelligence. Indeed, problems such as hyper-parameter optimization [7, 29], meta-learning [29], generative adversarial networks [30, 31], or robust neural network training [46] are naturally formalized with a bilevel structure.

Denote by x the upper-level variables and by y the lower-level variables. Denote by n_x and n_y the dimensions of x and y respectively. A bilevel problem is formulated in its general (optimistic) form by

$$\begin{aligned}
 & \min_{x \in X, y^* \in \mathbb{R}^{n_y}} && F(x, y^*) \\
 & \text{s. t.} && G(x, y^*) \leq 0 \\
 & && y^* \in \operatorname{argmin}_{y \in Y} f(x, y) \\
 & && \text{s. t.} \quad g(x, y) \leq 0,
 \end{aligned} \tag{1}$$

where $X \subseteq \mathbb{R}^{n_x}$ and $Y \subseteq \mathbb{R}^{n_y}$ represent the respective domains of x and y . The upper-level (UL) problem is represented by an objective function $F : X \times Y \rightarrow \mathbb{R}$ and constraints $G : X \times Y \rightarrow \mathbb{R}^{n_G}$, where n_G is the dimension of the upper-level (coupling) constraints. Conversely, the lower-level (LL) problem is described by its own objective function $f : X \times Y \rightarrow \mathbb{R}$ and constraints $g : X \times Y \rightarrow \mathbb{R}^{n_g}$, where n_g is the dimension of the lower-level constraints.

A BLO problem involves a hierarchical decision process where the upper-level decision-maker selects a variable $x \in X$. However, evaluating the upper-level objective F and constraints G requires anticipating the reaction of the lower-level decision-maker. Given a fixed x , the lower-level problem requires to minimize $f(x, y)$ subject to $y \in Y$ and $g(x, y) \leq 0$. The set of optimal solutions to this problem is not necessarily a singleton. In such cases, the upper-level may select any y among the lower-level optimal solutions for which the follower is indifferent. Among all such lower-level optimal solutions, the upper-level decision-maker selects one that satisfies the constraints $G(x, y) \leq 0$ and minimizes the objective $F(x, y)$. This setting is referred to as optimistic bilevel programming.

A standard approach to solve the BLO problem (1) consists in reformulating it as a single-level problem either by replacing the lower-level problem with its KKT optimality conditions [8] or by embedding the hyper-function associated with the lower-level value function (LLVF) [18, 52]. Penalization-based first-order methods built upon such reformulations have been shown to be effective [59]. Such methods, however, require first-order information or restrictive assumptions on the upper-level, which are not always available.

This work focuses on benchmarking techniques employed for comparing bilevel derivative-free optimization (BL-DFO) algorithms where either the upper-level or the lower-level problem in (1) is solved using a derivative-free optimization (DFO) method. The BL-DFO methods considered proceed iteratively by proposing an upper-level variable x and computing its associated optimal lower-level response y . Each iteration then involves solving the lower-level problem for a given x . A wide range of methods can be applied to solve the lower-level problem, either exactly or inexactly. Ensuring the optimality of lower-level responses y with respect to a given x is challenging and must be accounted for by benchmarking techniques in BLO.

1.1 Benchmarking techniques in BL-DFO

Benchmarking methodologies in DFO are well established for single-level DFO, but their extension to BLO is not trivial. A basic approach consists in reporting performance tables that provide raw data of optimization runs across solvers and test instances. Such tabular comparisons are used to support the performance of BL-DFO methods, e.g., [22, 38]. However, relying exclusively on tables may hinder interpretability, as extracting consistent conclusions from extensive tabulated results is cumbersome. The difficulty increases substantially when multiple solvers, runs, and test instances are considered. Convergence plots provide a graphical representation of the information reported in tables of statistics, namely the evolution of the value of the objective function with respect to the number of evaluations. They are widely used to compare BL-DFO algorithms (see e.g., [16] and references therein) to illustrate their performances on selected instances. Such a representation is appropriate when analyzing the performance of a given algorithm on a specific problem.

Although performance [23] and data [50] profiles are commonly used in DFO for benchmarking algorithms, their use remains limited in the BL-DFO literature. Islam et al. [34] assess the performance of their population-based DFO algorithm by multiple means, including performance profiles. In the context of direct search methods, Diouane et al. [20] use data profiles to compare both mesh-based and sufficient-decrease schemes, constructing the profiles solely using upper-level function evaluations. Recently, the line-search method devised by Cesaroni et al. [11] is benchmarked with performance and data profiles using lower-level evaluations as budget units.

The aforementioned benchmarking techniques are adapted to single-level DFO but do not account for issues arising from BLO. The lower-level optimality of produced solutions is not challenged by current benchmarking techniques. Yet, guaranteeing the feasibility and the lower-level optimality of solutions is mandatory for bilevel benchmarking techniques to ensure an equitable comparison between BL-DFO algorithms. Besides, the computational effort deployed to solve a bilevel problem is not always properly determined. The upper- and the lower-level efforts need to be adequately aggregated to represent the true computational cost deployed to solve (1).

1.2 Motivation and contributions

The main difficulty addressed in this paper arises when a DFO method claims that a point (x, y) is admissible for a bilevel problem, i.e., that it is feasible and that y is an optimal solution of the lower-level problem with respect to x . When the lower-level response is generated by a DFO algorithm, it is possible that it is not optimal for the lower-level due to local minima or an inability to converge. Therefore, that solution should have been discarded by the upper-level decision maker since non-admissible solutions cannot be considered optimal. Consequently, it is possible that an inefficient lower-level algorithm produces falsely admissible solutions. These solutions could satisfy the upper-level constraints and have an excellent upper-level objective function value. To summarize, this inefficient algorithm would claim to produce excellent solutions to the bilevel problem.

In this context, a procedure that challenges the admissibility of generated points, named *refereeing procedure*, is required to ensure a fair comparison of BLO algorithms in a DFO setting. The goal of such a procedure is to detect non-admissible points and remove them from the benchmarking data. Points where the lower-level responses are computed by an inefficient algorithm are therefore likely to be discarded. A referee takes as input the run log of an optimization run, analyzes entries that are claimed to be admissible, and flags the ones that are shown to be non-admissible. The revised log is then used by the benchmarking process.

Additionally, solving the lower-level problem for a given upper-level variable x may be computationally prohibitive. The computational effort devoted to this task by BL-DFO algorithms is not always considered by existing benchmarking techniques, while it represents a significant expense of derivative-free methods in solving a bilevel problem. Besides, applications considered may have upper-

and lower-level problems whose evaluation costs differ. A scaled computational effort that aggregates both upper- and lower-level evaluations allows then to account for the general cost involved in solving (1). This computational effort can then be incorporated into existing benchmarking techniques to measure the true expense employed to solve a bilevel problem.

The contributions of this work are summarized as follows.

- Notions about feasibility and admissibility in BL-DFO are extended to an inexact setting to cover practical issues. In addition, a review of existing BL-DFO algorithms is conducted, focusing on the inexactness of the methods employed to solve the lower-level problem and the benchmarking techniques used to compare such algorithms.
- A refereeing procedure is introduced to challenge the admissibility of points through additional optimization runs. Several strategies and settings are suggested to devise multiple refereeing strategies.
- A scaled computational effort that aggregates upper- and lower-level evaluations with a scaling factor is introduced. The combination of the scaled computational effort and the refereeing procedure yields an extension of single-level benchmarking techniques that accounts for the specific issues inherent to BLO.
- The proposed methodology is illustrated on a collection of analytical problems from the BOLIB library [61], adapted to the Julia programming language, and available as an open source package named `BOLIB.jl`.¹ The refereeing strategies are analyzed along with tests about the scaled computational effort to assess their impact on data profiles [50].

The paper is structured as follows. Section 2 explains how BLO can be considered in inexact and DFO settings. BLO notions are extended to the inexact setting, followed by an extensive literature review of BL-DFO algorithms that employ both exact and inexact methods to solve the lower-level problem. The refereeing procedure is detailed in Section 3 along with three refereeing strategies. Possible settings for the refereeing procedure are also discussed. The scaled computational effort is introduced in the description of the methodology as well. These features result in an extension of existing benchmarking techniques to BLO. Their impact is illustrated on numerical examples in Section 4.

Notations: $\mathbb{R}_+ := [0, +\infty)$ denotes the set of nonnegative real numbers, and $\overline{\mathbb{R}} := \mathbb{R} \cup \{\pm\infty\}$ represents the extended real numbers. For any integer $n \geq 1$, $\llbracket 1, n \rrbracket$ denotes the set $\{1, 2, \dots, n\}$ while $\mathbf{1}_n$ represents the vector of \mathbb{R}^n filled with 1.

2 Inexact bilevel optimization setting and DFO

BL-DFO methods studied in this work proceed iteratively by proposing an upper-level variable x , and then computing the associated lower-level response y . However, algorithms employed to solve the lower-level problem may produce inexact lower-level responses with respect to a tolerance ε . Setting $\varepsilon = 0$ means the lower-level problem is solved exactly. Additionally, bilevel optimization requires a dedicated terminology to characterize points that satisfy lower-level optimality and/or inequality constraints. This bilevel specific terminology is then extended to an inexact framework. A review of BL-DFO algorithms is presented to identify if the methods employed at the lower-level solve it exactly. Commonly used benchmarking techniques are also identified in this literature review and are further discussed.

2.1 BLO in an inexact setting

The specificity of bilevel structures require to define a specific terminology, adapted from [32], to characterize solutions produced by BL-DFO algorithms. This terminology also needs to be extended

¹<https://github.com/Vanadjy/BOLIB.jl>

to an inexact setting for situations in which lower-level responses associated to an upper-level variable x are computed inexactly.

A point $(x, y) \in X \times Y$ is said to be **feasible** for the nonlinear bilevel problem (1) if it satisfies the inequality constraints $G(x, y) \leq 0$ and $g(x, y) \leq 0$. The set of all feasible points is denoted by Ω . The set of optimal lower-level **responses** associated with an upper-level variable $x \in X$ is denoted by

$$\Psi(x) := \operatorname{argmin}_{y \in Y} \{f(x, y) : g(x, y) \leq 0\}.$$

A point $(x, y) \in X \times Y$ is said to be **rational** if y is an optimal lower-level response associated with x , i.e., $y \in \Psi(x)$. A point $(x, y) \in X \times Y$ is said to be **admissible** if $(x, y) \in \Omega$ and $y \in \Psi(x)$, i.e., (x, y) is feasible and rational. The set of all admissible points is called the **induced region**, denoted by \mathbb{IR} . Finally, a point $(x, y) \in X \times Y$ is **optimal** for (1) if $(x, y) \in \mathbb{IR}$ and

$$F(x, y) \leq F(\bar{x}, \bar{y}), \quad \forall (\bar{x}, \bar{y}) \in \mathbb{IR}.$$

BLO problems are inherently difficult. Even if all functions are linear, the induced region can be nonconvex and disconnected, and finding a global optimal solution is strongly NP-hard [32].

Some algorithms typically compute approximate minimizers of the lower-level optimization problem. To accommodate this inexactness, the notion of inexact lower-level optimality is formalized in Definition 1.

Definition 1. Let $\varepsilon = (\varepsilon_{obj}, \varepsilon_{\Omega}) \in \mathbb{R}_+^2$ be a tolerance vector for the bilevel problem (1). Consider an upper-level variable $x \in X$ and its associated lower-level response $y_{\varepsilon} \in Y$. Then, y_{ε} is said to be **ε -lower-level optimal** with respect to x if, for any $y \in Y$ that satisfies $g(x, y) \leq \varepsilon_{\Omega} \mathbf{1}_{n_g}$, the following inequalities hold

$$f(x, y_{\varepsilon}) \leq f(x, y) + \varepsilon_{obj} \quad \text{and} \quad g(x, y_{\varepsilon}) \leq \varepsilon_{\Omega} \mathbf{1}_{n_g}.$$

Denote by $\Psi_{\varepsilon}(x)$ the set of ε -lower-level optimal responses with respect to x .

Some BLO algorithms generate candidate points (x, y) that satisfy both upper-level and lower-level constraints G and g only up to a prescribed tolerance $\varepsilon_{\Omega} \geq 0$. All constraints are assumed to be scaled so that the same scalar tolerance ε_{Ω} can be used for all constraints; otherwise, a vector of nonnegative tolerances can be used instead. To account for inexact constraint satisfaction and inexact lower-level responses, the notions of feasibility, rationality, and admissibility are extended to the inexact setting in Definition 2

Definition 2. Let $\varepsilon = (\varepsilon_{obj}, \varepsilon_{\Omega}) \in \mathbb{R}_+^2$ be a tolerance vector for the bilevel problem (1). The point $(x, y) \in X \times Y$ is said to be

- **ε_{Ω} -feasible** if $G(x, y) \leq \varepsilon_{\Omega} \mathbf{1}_{n_G}$ and $g(x, y) \leq \varepsilon_{\Omega} \mathbf{1}_{n_g}$;
- **ε -rational** if y is ε -lower-level optimal with respect to x , i.e., $y \in \Psi_{\varepsilon}(x)$;
- **ε -admissible** if (x, y) is ε_{Ω} -feasible and ε -rational.

BL-DFO algorithms that employ inexact methods to compute lower-level responses are commonly used. Extending BLO terminology to the inexact setting therefore enables consideration of a relaxed criterion to revoke the admissibility of generated solutions. In this case, inexact methods used to solve the lower-level problem are not unfairly penalized in the benchmarking process.

2.2 A review of BL-DFO algorithms

Various benchmarking techniques are available to compare BL-DFO methods, but are often applied unevenly in the literature. This section provides an extensive literature review of existing BL-DFO methods to highlight the benchmarking techniques used in bilevel optimization. In addition to the benchmarking strategies, the review lists methods that solve the lower-level problem and mentions if it is assumed to be exact. All algorithms are classified based on the DFO scheme applied to the upper-level problem.

Direct search methods: A common approach for solving blackbox optimization problems is to exploit information obtained directly from objective function evaluations; hence the name direct-search methods [33]. These approaches either employ a mesh to select points and consider a simple decrease acceptance condition, or they consider a sufficient decrease criterion to accept new iterates [42]. More recently, Dzahini et al. [24] propose a unified review of the convergence analyses of direct search methods.

In the bilevel setting, Mersha and Dempe [49] developed a direct search framework along with a sufficient decrease condition. Two direct search variants for bilevel programming are proposed. The first requires that the set of poll directions has a sufficiently large vector density, while the second changes the set of poll directions for each unsuccessful iteration. The lower-level response is *a priori* given by an exact oracle for each upper-level variable x . Mersha and Dempe [49] proposed and analyzed directional direct-search methods in the BLO context; however, no implementation details for the proposed algorithms were provided. The first adaptation of a mesh adaptive direct-search [3] scheme was analyzed by Diouane et al. [20], where each lower-level response is obtained from an inexact oracle. They also studied direct-search strategies based on sufficient decrease in both smooth and nonsmooth settings. The performance of their methods is assessed using data profiles based on upper-level function evaluations.

For health insurance optimization problems, Zhang and Lin [60] developed direct search schemes to solve both upper-level and lower-level problems. The lower-level responses are obtained by applying a direct search scheme with sufficient decrease, and the tolerance for stopping the lower-level optimization is updated at each iteration. For one of their applications, the lower-level problem is solved using a variant of a trust-region method tailored to Nash-equilibrium problems [58]. They illustrate this approach with convergence plots on problems of hospital competitions [25] and oligopolistic market models.

Line-search methods: Recently, a BL-DFO framework was proposed by Cesaroni et al. [11], which includes a line search strategy performed by a projected extrapolation procedure that computes a direction d_k and a step length α_k . The upper-level variable is updated to decrease a hyper-objective function whose lower-level responses are computed with an inexact oracle. The performance of the proposed method is illustrated using tables, performance profiles, and data profiles.

Trust-region methods: Surrogates of the objective function using gradient and Hessian estimates can be constructed to apply trust-region methods in a derivative-free context. These approaches are adapted to problems where the derivatives exist but are not accessible. Colson et al. [13] adapted trust-region schemes to bilevel optimization, and Conn and Vicente [16] extended this framework to the DFO setting. They minimize quadratic interpolation models (minimum Frobenius norm models when not enough previously computed points are available) for both upper-level and lower-level problems. The lower-level problem is allowed to be solved inexactly, and the construction of models may exploit previously evaluated points. Their tests are conducted on analytical problems with linear box constraints for some of them. Convergence plots are drawn, showing the upper-level function value with respect to upper-level and lower-level function evaluations separately.

The algorithm proposed by Ehrhardt and Roberts [26] uses a trust-region scheme to solve the upper-level problem, while lower-level responses are obtained via an inexact oracle. They use either a gradient descent method or the Fast Iterative Shrinkage-Thresholding Algorithm (FISTA) [9] to solve a strongly convex lower-level problem arising from image denoising applications. The upper-level problem consists of a regularized non-linear least squares structure addressed in a DFO setting. This method is illustrated on mathematical imaging problems through convergence plots showing the upper-level value function with respect to lower-level evaluations.

Finite-differences methods: BL-DFO algorithms may exploit estimates of gradients or Hessians to drive the optimization process. Aghasi and Ghadimi [1] estimate an upper-level Hessian inverse and

partial gradient approximations using i.i.d. random vectors drawn from a standard Gaussian distribution. The lower-level problem is solved by a derivative-free stochastic gradient descent method, yielding an inexact lower-level response, which is then used in the outer optimization loop. The upper-level variables are also updated through a derivative-free projected stochastic gradient method. Computational tests are conducted on parameter optimization for classification models and display convergence plots with respect to both upper- and lower-level function evaluations separately.

An alternative stochastic DFO proximal-gradient framework, proposed by Staudigl et al. [55], allows both exact and inexact lower-level responses. Tests involve solving signal denoising and optimal experimental design problems. These are illustrated by error plots and other figures specific to their application. For transportation network problems, Maheshwari et al. [47] employ a derivative-free gradient descent to update the upper-level variables and a projected gradient descent to compute inexact lower-level responses. Numerical experiments are performed using convergence plots.

Bayesian optimization methods: Another approach to solve problems in DFO is based on the training of a Gaussian Process (GP) on a Design of Experiments (DoE). This approach, known as Bayesian Optimization (BO), consists in optimizing an acquisition function that yields trial points to enrich the DoE used to update the GP. Different strategies are employed for either exploration or intensification purposes with respect to the selected acquisition function and the values of its associated parameters.

To adapt BO to bilevel optimization, Ekmekcioglu et al. [27] design the algorithm BILBAO, which constructs a GP for both upper- and lower-level problems. The Regional Expected Value of Improvement constitutes the lower-level acquisition function and is optimized using a quasi-Newton method combined with a multi-start strategy. The resulting point (x, y) is used to update the lower-level GP, and a mapping function Φ is used to update upper-level variables, incurring inexact lower-level responses. Convergence plots related to the inexactness of lower-level responses are reported with respect to upper-level and lower-level function calls combined, assuming their evaluation costs are identical. Authors in [22, 38] present similar methods in which the upper-level variables are computed through the optimization of the lower confidence bound acquisition function. Lower-level responses are computed by a Sequential Least Squares Programming (SLSQP) method [40]. Most of the test results are presented in graphs and tables displaying fitness scores for both levels and the number of upper- and lower-level function evaluations.

To combine the advantages of several acquisition functions, Dogan and Prestwich [21] solve a multi-objective problem to suggest upper-level trial variables. The lower-level structure consists of a least squares problem and is solved using the SLSQP method. The convergence of the optimality gap is displayed: it shows the evolution of the gap between the best upper-level value found and a given optimal solution of the bilevel problem. The effort employed concerns upper-level evaluations (outer iterations) only.

Recently, the BILBO approach devised by Chew et al. [12] optimizes the Upper Confidence Bound (UCB) acquisition function to solve the upper-level problem. The lower-level function f is minimized through the UCB acquisition function only if f constitutes the largest regret bound. This latter is computed from an estimated lower-level optimal response on a trusted set of feasible solutions. The lower-level responses are supposedly computed exactly, but this can be extended to an inexact setting. Results are reported using convergence plots with respect to evaluations of any function involved in the bilevel problem, showing the decrease of the instant regret, which represents the loss associated with not selecting the optimal point. Based on an application of blackbox attacks on finite sequences, Man et al. [48] build a multi-kernel GP whose weights are updated by solving the upper-level problem block-wise. The acquisition function employed to solve the lower-level problem is the Expected Improvement, and the GP parameters are updated using maximum *a posteriori* estimations.

Heuristic-based methods: Efficient approaches in DFO are employed without convergence guarantees, known as heuristics. These methods can be used as standalone solvers or to enhance the performance

of existing algorithms in a DFO setting. Numerous existing BL-DFO algorithms employ population-based methods to solve the upper-level or the lower-level problems, notably genetic algorithms (GA).

The method developed by Jiang et al. [36] solves both upper- and lower-level problems by mutating populations of solutions using a simulated binary crossover and a polynomial mutation. A part of the upper- and lower-level populations is selected by a surrogate Kriging model, which approximates the relationship between upper-level variables and their associated lower-level responses. Lower-level optimization is performed by a GA with dynamic accuracy for each upper-level variable. Tables representing upper-level and lower-level accuracy, evaluation times, and euclidean distance to the best solution are used to assess the performance of their method.

The approach proposed by Ma et al. [45] addresses the upper-level problem with a Differential Evolution (DE) [17] strategy. The lower-level problem represents a surrogate learning stage performed by two successive gradient descent methods on two different loss functions, yielding a Kolmogorov-Arnold Network [43] used to approximate the lower-level objective function. Tables of mean scores and cost convergence plots with respect to lower-level evaluations are displayed to compare the different existing methods and their variants. The method devised by Islam et al. [34] uses a DE algorithm for the upper-level problem, while the lower-level function is optimized by another DE through three different surrogate models: linear and quadratic regressions, and Kriging models. Upper-level function values are displayed through tables and convergence plots for two different sets of instances [54] with respect to the total number of function evaluations (upper-level and lower-level combined with equivalent evaluation costs). Performance profiles are constructed from this set of instances, with an accuracy value defined by median accuracy and total function evaluations. The method for calculating the accuracy value is not mathematically explicit, and the admissibility of points is not discussed.

In a similar work of Islam et al. [35], a DE method is used for the upper-level framework, where each lower-level response is obtained from a BO algorithm [37]. An interior point algorithm is applied after BO to ensure the optimality of lower-level responses. The results of their tests are summarized in tables. Convergence plots showing the evolution of the upper-level function value with respect to upper-level and lower-level evaluations, equally combined, are also displayed. For a human gait application problem, Nguyen et al. [51] propose a GA framework for the upper-level problem, while a collocation strategy is used to compute lower-level responses by calling the IPOPT solver [57]. Their tests are illustrated by both tables showing solution errors and convergence plots over the number of generations of the GA.

2.3 Limits of existing benchmarking strategies in bilevel setting

The references above suggest extensions of single-level benchmarking techniques to a bilevel framework. To this end, the ε -admissibility of points generated by BL-DFO algorithms is assumed to be guaranteed. Yet, inexactly computed lower-level responses may result in non-admissible (and *a fortiori* non-optimal) solutions. Even claimed exact methods that solve the lower-level problem may not find a global optimal solution, as this would require expensive global optimization tools. Challenging the ε -admissibility of generated solutions is therefore required to extend benchmarking techniques to the bilevel setting to ensure fair comparisons between BL-DFO algorithms.

Existing benchmarking techniques account for either upper-level evaluations [16, 20, 21] or lower-level evaluations [11, 26, 44] as cost metrics. Counting upper-level evaluations alone disregards the computational effort involved in solving the lower-level problem, which constitutes a substantial cost in solving (1). Conversely, restricting the cost metric to lower-level evaluations is inadequate when evaluating the upper-level involves a significant expense. Both upper-level and lower-level evaluations are aggregated in some works in the literature (see [27, 34, 35]), assuming that upper- and lower-level evaluation costs are equivalent. Introducing a scaling factor that reflects the relative cost between the two levels is required to properly assess the total computational effort deployed to solve the bilevel problem.

Benchmarking techniques used to compare existing BL-DFO algorithms include tables, convergence plots, and benchmarking profiles, none of which fully address the difficulties arising from BLO.

- Tables reporting solving statistics become difficult to interpret at scale, making them unsuitable for systematic comparison across large algorithm and instance sets.
- Convergence plots illustrate the evolution of algorithms on individual instances and, thus, hardly support conclusions drawn over a broad collection of problem instances.
- Benchmarking profiles compare algorithms over a set of instances with respect to a prescribed effort metric, constituting the standard benchmarking technique in DFO. However, their use in BL-DFO remains scarce, as their constructions neither challenge the ε -admissibility of points nor account for the scaled aggregation of upper- and lower-level evaluations.

The information in this review about upper- and lower-level methods, the inexactness of lower-level responses, and the benchmarking techniques used is summarized in Table 1. The first column contains the source reference; the second indicates the upper-level algorithmic strategy. Then a checkmark indicates that the lower-level exact solutions can be obtained. The wide central column indicates the lower-level algorithmic strategy. Finally, the last three columns indicate whether the proposed analysis relies on tables, convergence plots, and/or profiles.

Table 1: Existing BL-DFO algorithms, with emphasis on upper-level (UL) and lower-level (LL) methodologies. The last three columns indicate if the analysis uses tables (T), convergence plots (C), or benchmarking profiles (P).

Reference	UL	Exact	LL Solver	T	C	P
Mersha and Dempe [49]	DS	✓	Unspecified			
Diouane et al. [20]	DS		Inexact Oracle			✓
Zhang and Lin [60]	DS		DFO method (direct search or trust-region)	✓	✓	
Cesaroni et al. [11]	LS		Inexact Oracle	✓		✓
Conn and Vicente [16]	TR		Trust-region method		✓	
Ehrhardt and Roberts [26]	TR		Gradient descent method or FISTA		✓	
Aghasi and Ghadimi [1]	FD		Gradient descent method		✓	
Staudigl et al. [55]	FD	✓	Exact and inexact Oracle			
Maheshwari et al. [47]	FD		Projected gradient method		✓	
Ekmekcioglu et al. [27]	BO		Acquisition function-based BO method		✓	
Dogan and Prestwich [22]	BO	✓	Nonlinear problem solved by SLSQP	✓		
Kieffer et al. [38]	BO	✓	Nonlinear problem solved by SLSQP	✓		
Dogan and Prestwich [21]	BO	✓	Least squares problem solved by SLSQP	✓	✓	
Chew et al. [12]	BO		Acquisition function-based BO method		✓	
Man et al. [48]	BO		Acquisition-function based BO method	✓	✓	
Jiang et al. [36]	EA		Population-based algorithm	✓		
Ma et al. [45]	EA		Gradient descent method	✓	✓	
Islam et al. [34]	EA		Surrogate-assisted differential evolution method	✓	✓	✓
Islam et al. [35]	EA	✓	Two phases: a BO method then an interior point method	✓	✓	
Nguyen et al. [51]	EA	✓	Direct collocation method	✓	✓	

This review of BL-DFO methods motivates the need to generalize the methodology of current benchmarking techniques to the bilevel setting. To this end, a refereeing procedure is introduced to describe how to challenge the ε -admissibility of points, and a scaling method is proposed to properly aggregate upper-level and lower-level evaluations into a unified effort metric.

3 A referee-based bilevel benchmarking technique

Current benchmarking techniques do not adequately consider the following BLO issues. Firstly, the ε -admissibility of solutions produced by a BL-DFO algorithm is not always guaranteed. Challenging the ε -admissibility of points through a procedure that discards non-admissible points is therefore required. Secondly, upper- or lower-level evaluations are either ignored or not properly aggregated. A scaled computational effort that accounts for the actual expenses at both levels is therefore needed to reflect the general cost to solve problem (1). A referee-based procedure is introduced along with refereeing

strategies to challenge the ε -admissibility of solutions. Possible settings for the refereeing procedure are then detailed. Finally, a scaled computational effort is introduced and the evaluation budget units used in data profiles are discussed to extend this benchmarking technique to BLO.

3.1 A refereeing procedure to revoke admissibility

A procedure for benchmarking bilevel algorithms is required to challenge the claimed ε -admissibility of points generated by a BL-DFO algorithm, named *refereeing procedure*, along with strategies to select which points are challenged. A criterion of non-admissibility is first given to revoke solutions that are not ε -admissible before detailing the procedure. For a given upper-level variable $x \in X$, the corresponding lower-level response $y \in Y$ may be suboptimal, as the lower-level algorithm might terminate prematurely, converge to a local minimizer, or simply fail. In blackbox, nonsmooth or nonconvex settings, global optimality is generally unattainable. The ε -admissibility of an ε_Ω -feasible point $(x, y) \in X \times Y$ is revoked when an alternative ε_Ω -feasible point $(x, \hat{y}) \in X \times Y$ is shown to satisfy

$$f(x, \hat{y}) < f(x, y) - \varepsilon_{obj}.$$

Consider a benchmark set of instances \mathcal{P} and a collection of algorithms \mathcal{A} . The instances of \mathcal{P} should be chosen to be as general as possible by avoiding biased initial conditions or artificial structures. Additionally, it should be representative of the class of target problems of interest. Let $\mathcal{H}_{a,p}$ be the set of claimed ε -admissible points generated by applying algorithm $a \in \mathcal{A}$ on the problem instance $p \in \mathcal{P}$. Let $H \in \mathbb{N}$ be the cardinality of $\mathcal{H}_{a,p}$, so that this set can be written as

$$\mathcal{H}_{a,p} := \{(x_k, y_k) : k \in \llbracket 0, H \rrbracket\}. \quad (2)$$

Definition 3 describes the *a posteriori* procedure to challenge, and eventually to revoke, the ε -admissibility of a point of $\mathcal{H}_{a,p}$.

Definition 3. Let $\varepsilon = (\varepsilon_{obj}, \varepsilon_\Omega) \in \mathbb{R}_+^2$ be a tolerance vector and $(x, y) \in \mathcal{H}_{a,p}$, an ε_Ω -feasible point for problem instance p produced by the BLO algorithm a . A **referee** is an algorithm r that **challenges** the ε -admissibility of (x, y) through the following steps:

1. Apply the referee r with upper-level variable x to produce a lower-level response y^r ;
2. If (x, y^r) is ε_Ω -feasible and if $f(x, y^r) < f(x, y) - \varepsilon_{obj}$, then the ε -admissibility of (x, y) is **revoked**.

Each referee can belong to \mathcal{A} (Internal Referee) or not (External Referee). A referee does not certify the admissibility of a point (x, y) . Instead, it challenges possibly non-admissible points by looking for an alternative lower-level response that contradicts the presumed ε -lower-level optimality of the response y for the same x . The refereeing procedure is intended to be applied as a preprocessing step to the benchmarking process.

Let r be a referee. The refereeing procedure may be specified differently depending on the purposes of the optimization framework. Accordingly, three distinct refereeing strategies are defined, each tailored to a specific validation criterion, yielding a filtered history $\mathcal{H}_{a,p}^r \subseteq \mathcal{H}_{a,p}$.

- The End-Point Referee challenges only the ε -admissibility of the final point (x_H, y_H) .
- The Complete Referee challenges the ε -admissibility of all points (x_k, y_k) for $k \in \llbracket 0, H \rrbracket$.
- The Reverse Referee challenges the ε -admissibility of some points in reverse order, starting from the final point (x_H, y_H) .

The End-Point Referee challenges only the last point of $\mathcal{H}_{a,p}$, namely (x_H, y_H) . If the ε -admissibility of (x_H, y_H) is revoked, then the entire execution history $\mathcal{H}_{a,p}$ is discarded. Otherwise, all the points (x_k, y_k) for $k \in \llbracket 0, H \rrbracket$ are preserved in $\mathcal{H}_{a,p}^r$. Then, for the End-Point strategy,

$$\mathcal{H}_{a,p}^r = \begin{cases} \mathcal{H}_{a,p} & \text{if } (x_H, y_H) \text{ is not revoked by } r, \\ \emptyset & \text{otherwise.} \end{cases}$$

This strategy disregards the trajectory of the algorithms and bases its decision solely on the ε -admissibility of the returned solution. The End-Point Referee is inherently myopic, as it may discard points that might be ε -admissible simply because the last point is non-admissible.

The Complete Referee challenges each point $(x_k, y_k) \in \mathcal{H}_{a,p}$ and discards every revoked point from $\mathcal{H}_{a,p}$ yielding the corresponding filtered history $\mathcal{H}_{a,p}^r$, which represents $\mathcal{H}_{a,p}$ deprived of the points revoked by r , i.e.,

$$\mathcal{H}_{a,p}^r = \{(x_k, y_k) \in \mathcal{H}_{a,p} : k \in \llbracket 0, H \rrbracket \text{ and } (x_k, y_k) \text{ is not revoked by } r\}.$$

The resulting filtered history $\mathcal{H}_{a,p}^r$ can then be used to construct performance and data profiles [23, 50]. However, the Complete Referee is computationally expensive, as it requires rerunning multiple optimization procedures for each history $\mathcal{H}_{a,p}$.

The Reverse Referee is designed to mitigate the drawbacks of the End-Point and the Complete Referees. It challenges the ε -admissibility of the last point (x_H, y_H) first, then proceeds backward through $\mathcal{H}_{a,p}$ until either an ε -admissible point is identified, denoted by (x_K, y_K) , or the ε -admissibility of the point (x_0, y_0) is revoked. If such an integer K exists, then all points (x_k, y_k) for $k \in \llbracket 0, K \rrbracket$ are preserved, thus

$$\mathcal{H}_{a,p}^r = \left\{ \begin{array}{l} (x_k, y_k) \in \mathcal{H}_{a,p} : k \in \llbracket 0, K \rrbracket \text{ where } K \in \llbracket 0, H \rrbracket \text{ satisfies} \\ \quad (x_K, y_K) \text{ is not revoked by } r, \\ \quad (x_\ell, y_\ell) \text{ is revoked by } r \text{ for all } \ell \in \llbracket K + 1, H \rrbracket \end{array} \right\}.$$

Otherwise, all points of $\mathcal{H}_{a,p}$ are revoked, resulting in $\mathcal{H}_{a,p}^r = \emptyset$.

If the refereeing procedure is deterministic, then both the Complete and Reverse referees identify the same final ε -admissible solution. The Reverse Referee is therefore well-suited for constructing accuracy profiles [10], as it provides the final ε -admissible solution for each algorithm-instance point and avoids verifying the entire history.

Several algorithms may be used for a single refereeing procedure. Denote by \mathcal{R} the set of referees selected to challenge the ε -admissibility of points of $\mathcal{H}_{a,p}$. The filtered history resulting from this refereeing procedure is denoted by

$$\mathcal{H}_{a,p}^{\mathcal{R}} := \bigcap_{r \in \mathcal{R}} \mathcal{H}_{a,p}^r. \quad (3)$$

Refereeing strategies differ in computational cost as they do not challenge the same number of points of $\mathcal{H}_{a,p}$. The End-Point Referee is the least expensive strategy since it challenges only the last point generated during the run of algorithm a on instance p . Its overall cost is therefore proportional to the number of instances $|\mathcal{P}|$. The Complete Referee challenges the ε -admissibility of all points in $\mathcal{H}_{a,p}$, so the cost of this refereeing process scales with the length of each execution history. The Reverse Referee has a lower computational cost than the Complete Referee, as it challenges the ε -admissibility on a subset of $\mathcal{H}_{a,p}$, proceeding backward from the final point (x_H, y_H) . In the worst case scenario, i.e., when the ε -admissibility of each point is revoked, its computational cost coincides with that of the Complete Referee. Conversely, the Reverse Referee cost is identical to that of the End-Point Referee in the best case scenario, i.e., when the ε -admissibility of all (x_H, y_H) is not revoked. If there are multiple referees, the computational cost for the benchmarking can exceed that of the optimization process.

3.2 Referees settings

The refereeing procedure is sensitive to the configuration of the referees. The choice of the starting point for the lower-level re-optimization critically influences the outcome of the refereeing procedure. Likewise, the input parameters supplied to the referee directly condition the lower-level optimization process.

Initialization of the starting point: A critical design choice in the refereeing procedure concerns the initialization of the referee r when it challenges a point (x_k, y_k) produced by algorithm a . A first approach is to systematically select the same starting point as the one employed by algorithm a to produce a lower-level response. In other words, if a is initialized at y_k^0 and produces the solution (x_k, y_k) , r is initialized at y_k^0 as well. A straightforward way to apply this strategy is to use the previously computed lower-level response, i.e., set $y_k^0 = y_{k-1}$. To this end, algorithm a needs to track all the initial points it used to generate each lower-level response. Nevertheless, it enables the referee r to be in similar conditions as the lower-level algorithm employed by a in the original optimization process.

As a second approach, initializing the referee at y_k promotes an intensified local search in the neighborhood of the challenged point. Notice that the total function evaluation budget is frequently exhausted during the original run in a DFO setting. Consequently, this approach may revoke a large proportion of challenged points, possibly yielding $\mathcal{H}_{a,p}^{\mathcal{R}} = \emptyset$ regardless of the true quality of the benchmarked algorithms.

A third approach can be applied when the problem instance $p \in \mathcal{P}$ supplies a nominal starting point (x_0, y_0) , the referee can be initialized at y_0 for all challenged points. This helps mitigate r being entrapped in the same local minima as a , allowing for the possibility of computing better alternative lower-level solutions by looking for unexplored feasible regions.

Parameters of referees: Settings differences between the referee and the algorithms in \mathcal{A} directly affect the revocation of challenged points, specifically parameters related to stopping conditions. Allocating a larger evaluation budget to the referee might identify better lower-level solutions at the cost of additional computational expenses. Conversely, reducing the budget or relaxing termination tolerances lowers the computational overhead of the refereeing procedure but compromises the ε -admissibility verification.

Method-specific parameters can critically affect the outcome of the refereeing procedure, e.g., the mesh shrinkage parameter for mesh-based direct search methods or the initial trust-region radius of trust-region methods. Adjusting parameters differently between an internal referee r and a might result in unexpected outcomes in the refereeing procedure. A straightforward choice in the setting of an internal referee is to select all of its parameters to be identical to those selected for a .

The chosen settings for a given referee r need to remain identical for all algorithms $a \in \mathcal{A}$ to ensure a fair refereeing procedure. Furthermore, other possible refereeing settings may be used in addition to the aforementioned ones to modulate the outcomes of the refereeing procedure. These settings can also be combined to calibrate the refereeing procedure to user-specific requirements.

3.3 Computational effort and evaluation budget unit

For a fixed upper-level variable, generating its associated lower-level response entails multiple evaluations of the lower-level problem, which constitutes a substantial computational expense in solving (1). Besides, upper- and lower-level evaluation costs might differ in a blackbox optimization context. For these reasons, one can consider scaling both upper- and lower-level evaluations to account for their relative computational costs.

Consider a scaled computational effort that aggregates upper-level and lower-level function evaluations as

$$N := \lambda N_{UL} + N_{LL}, \quad (4)$$

where N_{UL} and N_{LL} represent the numbers of evaluations of the upper-level and the lower-level problems, respectively. The scaling parameter $\lambda \geq 0$ represents the relative cost of evaluating the upper-level and the lower-level. This aggregation is similar to the one proposed in a surrogate-assisted framework [6].

When the upper-level evaluation cost is insignificant compared that of the lower-level, setting $\lambda = 0$ enables to account for lower-level evaluations only. One has $\lambda \geq 1$ if the upper-level is more onerous than the lower-level, and $\lambda = 1$ implies that evaluating both levels involves the same computational effort, as assumed in [27, 34, 35]. The value of λ can be determined using information related to the instances, e.g., by computing the ratio t_{UL}/t_{LL} where t_{UL} and t_{LL} are the elapsed CPU times to evaluate the upper-level and the lower-level, respectively. Otherwise, λ can be fixed by the user to enable adaptability to instance-specific characteristics.

An alternative scaled computational effort can be defined by

$$N := N_{UL} + \lambda^{-1}N_{LL}. \quad (5)$$

Both (4) and (5) provide measures of the overall computational cost involved by a bilevel algorithm. The scaling factor λ or λ^{-1} is used for indicating whether the upper- or the lower-level problem is the most costly to be evaluated. Formulation (4) is adapted to situations in which evaluating the upper-level problem is more onerous than evaluating the lower-level. Conversely, formulation (5) would be preferred.

The scaled computational effort N , with either formulation (4) or (5), accounts for the efforts required at both levels. However, the dimensions of both problems remain to be considered by groups of budget units for data profiles to extend them to BLO. Data profiles constructed by Diouane et al. [20] employ groups of $n_x + 1$ evaluations, counting solely upper-level evaluations, which do not properly represent the global effort deployed to solve (1). Cesaroni et al. [11] constructed data profiles using groups of $n_x n_y + 1$ lower-level function evaluations as effort scaling. This approach accounts for the dimensions of both problems involved in (1), but does not enable a straightforward generalization of data profiles: for single-level problems, $n_y = 0$ resulting in $n_x n_y + 1 = 1$.

Groups of $(n_x + 1)(n_y + 1)$ budget units are proposed to extend data profile evaluation groups to BLO for two reasons. The first one is that it naturally generalizes the groups of $n_p + 1$ evaluations proposed by Moré and Wild [50] for single-level optimization, where n_p is the dimension of problem instance p . Indeed, if there is no lower-level problem (i.e., $n_y = 0$), then the groups considered are of $n_x + 1$ (upper-level) evaluations, which corresponds exactly to groups of $n_p + 1$ evaluations. The second reason is that $(n_x + 1)(n_y + 1)$ represents the minimal number of iterations required by a single unsuccessful step in a bilevel direct search poll phase. In other words, the minimal effort deployed by a BL-DFO algorithm employing a direct search scheme at both levels to terminate the optimization of problem (1) is of $(n_x + 1)(n_y + 1)$ evaluations. Such groups, therefore, represent the same minimal budget required by a single-level direct-search algorithm to terminate a poll phase with a minimal positive basis, i.e., $n_p + 1$ evaluations. Otherwise, existing benchmarking techniques remain unchanged, with the filtered data $\mathcal{H}_{a,p}^R$ and the scaled effort N used in place of their single-level counterparts. The refereeing procedure challenges the admissibility of points to discard non-optimal solutions, while the scaled computational effort N more accurately reflects the true computational cost to solve (1).

4 Numerical illustrations

In this section, the proposed refereeing strategies, as well as the scaled computational effort, are illustrated on a set of bilevel problems in order to assess their impact on existing benchmarking techniques commonly used in DFO. The set of instances is taken from the BOLIB Matlab library [61], which consists of analytical bilevel problems. Except for the internal refereeing strategy, in which the identified DFO solvers are explicitly cited, the solvers are anonymized as `Algo1`, `Algo2`, and `Algo3` throughout the tests.

4.1 Implementation details

The refereeing strategies and tested solvers are all implemented in the Julia 1.11 programming language. For this work, a Julia variant of the BOLIB [61] library, named `BOLIB.jl`, is developed to make

numerous bilevel problems available in open source. `BOLIB.jl` contains a majority of BOLIB instances under the `BilevelProblem` mutable structure. Tests are conducted on a standard desktop computer with 64 GB of RAM.

The initial point (x_0, y_0) corresponds to the one given in the BOLIB instance definition. The budgets of function evaluations are $N_{UL}^{tot} = 300$ and $N_{LL}^{tot} = 100n_y$ for the upper-level and the lower-level, respectively. The tolerances used in the refereeing procedures to revoke the ε -admissibility of points is set as $\varepsilon_{obj} = 10^{-9}$ and $\varepsilon_{\Omega} = 0$. A two-phase extreme barrier scheme [4, Chapter 12] is used in the situations in which the starting point is infeasible.

The End-Point, Complete and Reverse refereeing procedures are compared under specific settings. The initialization of each challenged point uses the lower-level variable y_0 provided by the BOLIB instance. Each refereeing procedure uses the same evaluation budget and stopping tolerance that were used by the algorithms of \mathcal{A} for the lower-level problem. All other algorithmic parameters are set to the default values, except when specific setting modifications are explicitly mentioned.

The value of the scaling parameter λ is straightforwardly set as $\lambda := \frac{t_{UL}}{t_{LL}}$, where t_{UL} and t_{LL} are the CPU times to call the upper-level and the lower-level functions, respectively. A different value of λ is used in Section 4.2 to specifically illustrate its impact. Since the computational expenses to evaluate the upper- and the lower-level problems are equivalent for a majority of BOLIB instances, formulation (4) of the scaled computational effort N is chosen for all instances.

The accuracy value [6] used to construct the profiles is defined by $F_{acc}^N := \frac{F^N - F^0}{F^* - F^0}$, where F^N is the value of F after a deployed effort of N effort units, F^0 is the largest initial ε -admissible value of F and F^* is the best ε -admissible value of F found among all solvers. The precision values $\tau \in \{10^{-1}, 10^{-2}, 10^{-4}\}$ are used to determine whether an instance is τ -solved (i.e., $F_{acc}^N \geq 1 - \tau$) or not. The percentage of τ -solved instances is displayed on the y -axis of the data profiles.

4.2 Impact of scaling the computational effort

The first test set illustrates the impact of employing $N = \lambda N_{UL} + N_{LL}$ against $N = N_{UL}$ and $N = N_{LL}$. The context is a situation where evaluating the upper-level is significantly more expensive than evaluating the lower-level: the parameter λ is set to 60 for problem instances. Two anonymous algorithms with different lower-level subsolvers, named `Algo1` and `Algo2`, are compared and the results are illustrated with data profiles in Figure 1 on 67 BOLIB instances. The central plot uses the scaled computational effort $N = \lambda N_{UL} + N_{LL}$ to represent the actual cost spent to solve (1). The data profiles on the left considers only the upper-level function evaluations (i.e., $N = N_{UL}$) while the plot on the right considers only the lower-level function evaluations (i.e., $N = N_{LL}$).

Notice that the groups of budget units on the x-axis in Figure 1 are not identical, resulting in different axis labels. For $N = N_{UL}$, the effort measured relies only on upper-level evaluations. In this case, to adapt the x-axis for the left data profile, evaluations are measured in groups of $(n_x + 1)$ UL evaluations. Conversely, with $N = N_{LL}$, the effort measured relies only on the LL evaluations; therefore, groups of $(n_y + 1)$ LL evaluations are considered. The scaled computational effort N is defined by (4) and groups of $(n_x + 1)(n_y + 1)$ are considered, as detailed in Section 3.3. The budget unit for the data profiles in the central column refers to lower-level equivalent evaluations, as N defined by formulation (4) quantifies the deployed effort from a lower-level perspective.

The central plot suggests that `Algo2` outperforms `Algo1`. However, considering solely upper-level evaluations artificially improves the performances of `Algo1`, resulting in a data profile equivalent to that of `Algo2`. Conversely, considering solely lower-level evaluations involves a larger gap between the data profiles of `Algo1` and `Algo2` than those where $N = \lambda N_{UL} + N_{LL}$ is considered. `Algo2` therefore dominates `Algo1` if solely lower-level evaluations are counted. Consequently, neglecting either the upper- or the lower-level effort may result in inaccurate comparisons between benchmarked algorithms.

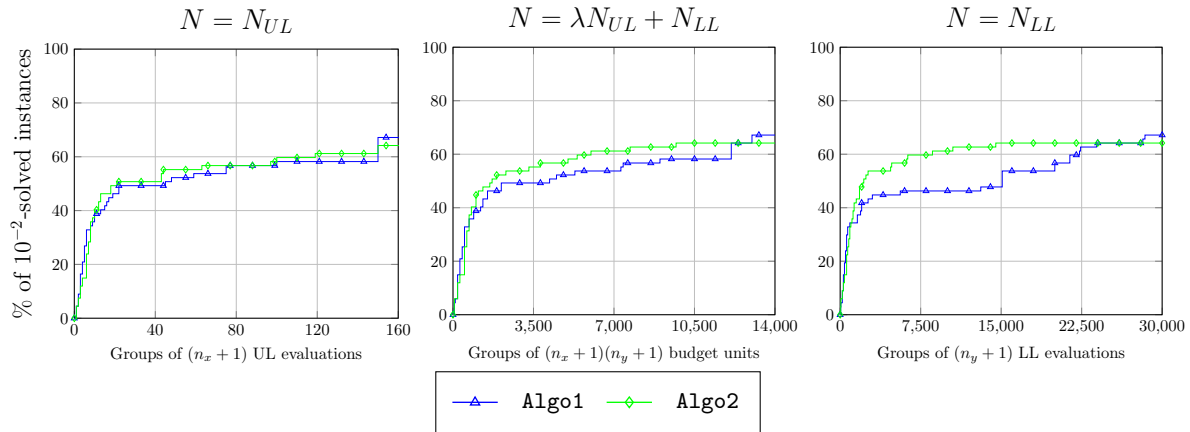


Figure 1: Data profiles considering as effort metric solely upper-level evaluations N_{UL} (left), a scaled computational effort $N = \lambda N_{UL} + N_{LL}$ with $\lambda = 60$ (middle) and solely lower-level evaluations N_{LL} (right) on 67 BOLIB instances.

4.3 External referee

The following example demonstrates that omitting the refereeing procedure can lead to erroneous conclusions. Three solvers named **Algo1**, **Algo2**, and **Algo3**, differing only in the employed subsolver to address the lower-level problem, are compared. For each variant, an External Referee is used to solve the lower-level problem and challenges the solutions generated by employing the End-Point, Reverse, and Complete refereeing strategies.

Results are reported by data profiles in Figure 2. The plots in the left column compare the algorithms without any refereeing procedure for two values of τ . In other words, it is assumed that all solutions produced by the algorithms are ε -admissible. The three other columns involve a referee that challenges the ε -admissibility of some solutions. The effect is drastic: the ε -admissibility of many solutions produced by **Algo1** and **Algo3** is revoked by the referee. As a consequence, the data profiles of these two algorithms are lowered, which benefits the performance of **Algo2**. With any of the three refereeing processes, **Algo2** is clearly shown to be dominant.

Figure 2 also suggests that the outcomes of the Reverse Referee and of the Complete Referee are similar, as the third and fourth columns of the figure are comparable. However, in this example the inexpensive End-point Referee shows **Algo1** dominates **Algo3**. The Reverse and Complete Referees show instead that **Algo1** and **Algo3** are comparable with both precisions $\tau = 10^{-1}$ and $\tau = 10^{-4}$.

The computational time required by the three strategies to complete the refereeing procedures are

End-Point: 2.0 minutes, Reverse: 1.9 hours, Complete: 3.6 hours.

Obviously, the End-Point Referee completes the refereeing procedure in a substantial lower time than the other refereeing strategies. The Reverse Referee requires slightly less than 50% of the time needed by the Complete Referee while producing similar data profiles. It is therefore the most economical refereeing strategy for generating benchmarking profiles comparable to those of the Complete Referee.

In summary, applying any of the refereeing processes, even the inexpensive referee that challenges a single point, is necessary to benchmark the algorithms. However, more expensive refereeing processes are required for a finer analysis. The Reverse Referee is preferable in this example, as it produces similar conclusions to those from the Complete Referee, with lower computational effort.

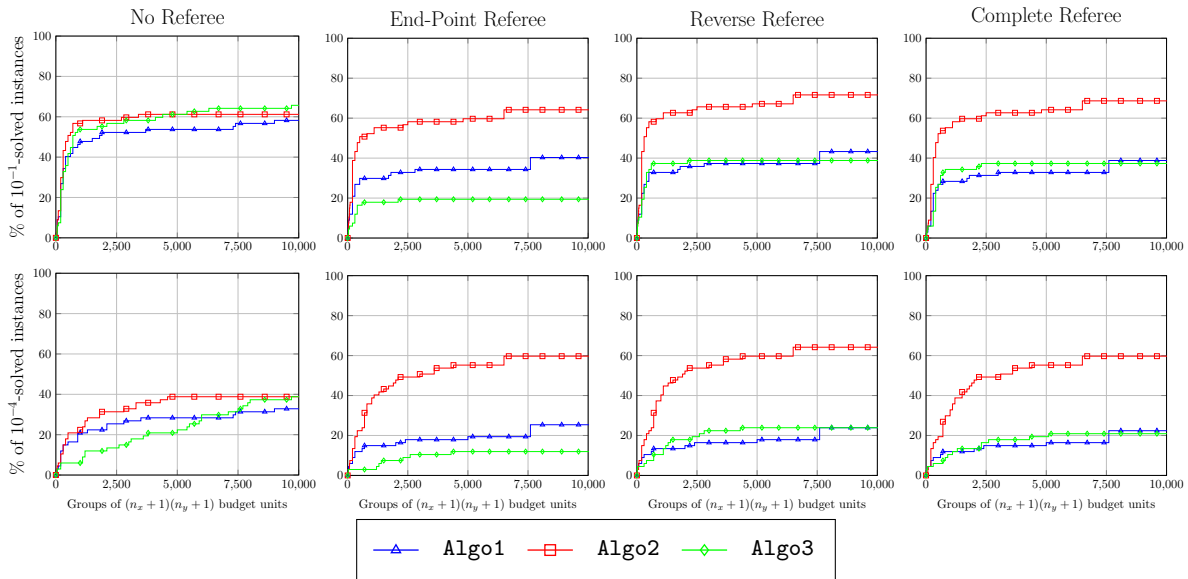


Figure 2: Data profiles employing (from left to right) No Referee, End-Point, Reverse and Complete refereeing strategies carried out using an External Referee with a tolerance $\varepsilon = (10^{-9}, 0)$ to revoke ε -admissibility on 67 BOLIB instances.

4.4 Internal referee

The last example shows the effect of the refereeing procedure on algorithms employing the same upper-level solver with different lower-level solvers. The upper-level simply uses the Mesh Adaptive Direct Search (MADS) [3] poll step. The lower-level solvers are three similar direct search algorithms implemented in the NOMAD software package [5]. The first one is a Coordinate Search (CS) algorithm [28]. The second one is MADS in which the search step [15] is disabled. MADS is an upgrade of CS. The third one is the most sophisticated algorithm, as it is the default implementation of MADS in NOMAD with the quadratic search step [15]. These last two algorithms are denoted by MADS No Search and MADS Quad Search. One would expect that the profiles would rank the algorithms from least to most efficient as CS, MADS No Search and MADS Quad Search.

The results are illustrated by data profiles in Figure 3. As with Figure 2, the data profiles need to be read from left to right. The first column compares the algorithms without any refereeing procedure, i.e., assuming all the solutions produced by the algorithms are ε -admissible. Each other column applies a refereeing strategy to challenge the ε -admissibility of solutions. The three algorithms appear to be either comparable, or ranked as MADS No Search slightly better than MADS Quad Search better than CS when no referee is applied.

In the three columns where the refereeing process is applied, the data profile of CS is drastically lowered, while those of MADS No Search and MADS Quad Search are less impacted. The expected hierarchy between the three direct search methods is exhibited when any refereeing strategy is applied: MADS Quad Search is an improvement on MADS No Search, which is an improvement on CS.

Some differences are noticeable between the three refereeing strategies illustrated in Figure 3. MADS No Search and MADS Quad Search revoke the admissibility of many solutions that were claimed ε -admissible by CS. The Reverse Referee slightly increases the data profiles of some algorithms compared to the Complete Referee. The general conclusions drawn from Figure 3 are the same regardless of the applied refereeing strategy.

The computational time required by the three strategies to complete the refereeing procedures are

End-Point: 1.2 minutes, Reverse: 20.9 minutes, Complete: 2.1 hours.

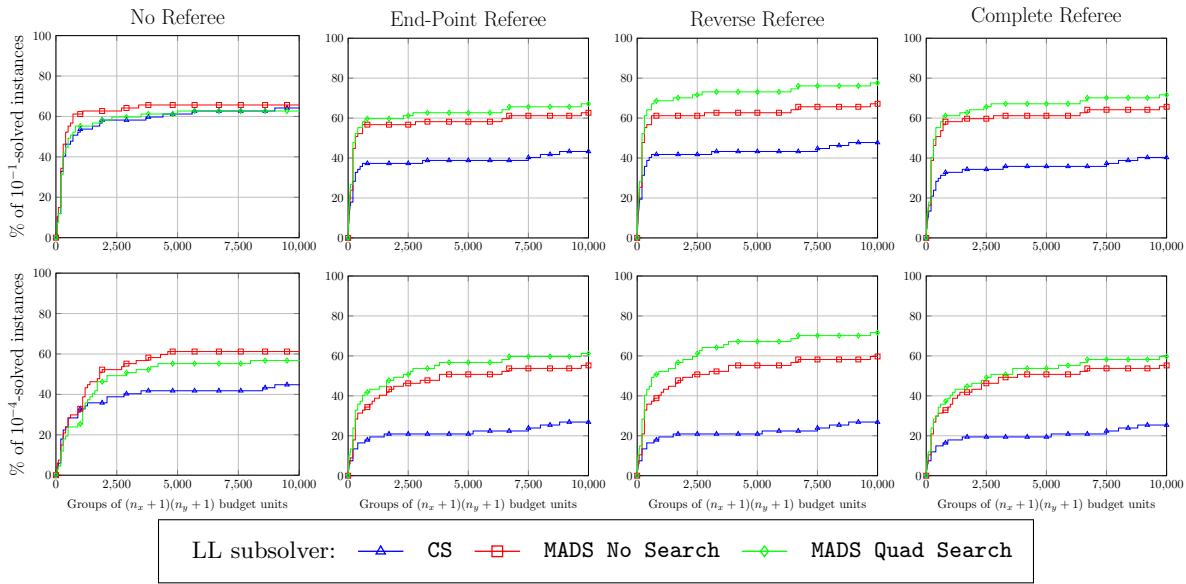


Figure 3: Data profiles employing (from left to right) No Referee and Internal End-Point, Reverse and Complete refereeing strategies with a tolerance $\varepsilon = 10^{-9}$ to revoke the ε -admissibility on 67 BOLIB instances.

The End-Point Referee remains the cheapest strategy by far as it is 20 times faster than the Reverse Referee and 100 times faster than the Complete Referee. Similar conclusions as in Section 4.3 can be drawn about the CPU times required by the different refereeing strategies.

In summary, applying any refereeing procedure, even the less expensive one, on similar direct search methods enables to recover an established algorithms hierarchy. In this example, the End-Point referee is the most adapted strategy as it results in the most accurate data profiles compared to the Complete Referee for the least computational expense. The Reverse Referee slightly overestimates the performances of algorithms, but without impacting the general conclusions drawn from the data profiles.

5 Conclusion

This work introduces an extension of existing single-level DFO benchmarking techniques to the bilevel setting. A refereeing procedure is proposed to challenge the ε -admissibility of solutions generated by BL-DFO algorithms through three strategies: the End-Point, Reverse, and Complete Referees. Each strategy involve an additional computational cost to remove non-admissible points from the logs produced by the algorithms. The End-Point Referee is inexpensive, but less reliable than the other two strategies. The Reverse Referee is our recommended approach, as it leads to conclusions similar to those of the Complete Referee while requiring a least computational cost.

Additionally, a scaled computational effort that aggregates both upper- and lower-level evaluations is introduced to better represent the overall effort required to solve nonlinear bilevel problems. Both features are illustrated on analytical test problems. The refereeing procedure is shown to discard non-admissible points generated by a BL-DFO algorithm, thereby yielding more faithful conclusions regarding algorithmic performance, albeit at a substantial additional computational cost. Similarly, the scaled computational effort provides a more accurate measure of the total effort deployed to solve a bilevel problem. Neglecting part of this effort may lead to overestimated performances in data profiles.

References

- [1] A. Aghasi and S. Ghadimi. [Fully zeroth-order bilevel programming via Gaussian smoothing](#). *Journal of Optimization Theory and Applications*, 205(2):31, 2025.
- [2] A. Aghasi, I. Mendoza-Sanchez, L. Abriola, and E. Miller. [Joint electrical and hydrological inversion for reconstruction of subsurface contaminant source zones](#). In *IEEE International Geoscience and Remote Sensing Symposium*, pages 7059–7062, 2012.
- [3] C. Audet and J. Dennis, Jr. [Mesh Adaptive Direct Search Algorithms for Constrained Optimization](#). *SIAM Journal on Optimization*, 17(1):188–217, 2006.
- [4] C. Audet and W. Hare. [Derivative-Free and Blackbox Optimization](#). Springer Series in Operations Research and Financial Engineering. Springer, Cham, Switzerland, 2017.
- [5] C. Audet, S. Le Digabel, V. Rochon Montplaisir, and C. Tribes. [Algorithm 1027: NOMAD version 4: Nonlinear optimization with the MADS algorithm](#). *ACM Transactions on Mathematical Software*, 48(3): 35:1–35:22, 2022.
- [6] C. Audet, W. Hare, and C. Tribes. [A summary of benchmarking constrained, multi-objective and surrogate-assisted derivative-free optimization methods](#). *Optimization Letters*, 2026. In press.
- [7] F. Bao, G. Wu, C. Li, J. Zhu, and B. Zhang. [Stability and generalization of bilevel programming in hyperparameter optimization](#). *Advances in neural information processing systems*, 34:4529–4541, 2021.
- [8] J. Bard. [An algorithm for solving the general bilevel programming problem](#). *Mathematics of Operations research*, 8(2):260–272, 1983.
- [9] A. Beck and M. Teboulle. [A fast iterative shrinkage-thresholding algorithm for linear inverse problems](#). *SIAM journal on imaging sciences*, 2(1):183–202, 2009.
- [10] V. Beiranvand, W. Hare, and Y. Lucet. [Best practices for comparing optimization algorithms](#). *Optimization and Engineering*, 18(4):815–848, 2017.
- [11] E. Cesaroni, G. Liuzzi, and S. Lucidi. [Derivative-Free Bilevel Optimization with Inexact Lower-Level Solutions](#). Technical Report 2603.20759, arXiv, 2026.
- [12] R. Chew, Q. Nguyen, and B. Low. [BILBO: BILevel Bayesian Optimization](#). Technical Report 2502.02121, arXiv, 2025.
- [13] B. Colson, P. Marcotte, and G. Savard. [A trust-region method for nonlinear bilevel programming: algorithm and computational experience](#). *Computational Optimization and Applications*, 30(3):211–227, 2005.
- [14] B. Colson, P. Marcotte, and G. Savard. [An overview of bilevel optimization](#). *Annals of Operations Research*, 153(1):235–256, 2007.
- [15] A. Conn and S. Le Digabel. [Use of quadratic models with mesh-adaptive direct search for constrained black box optimization](#). *Optimization Methods and Software*, 28(1):139–158, 2013.
- [16] A. Conn and L. Vicente. [Bilevel derivative-free optimization and its application to robust optimization](#). *Optimization Methods and Software*, 27(3):561–577, 2012.
- [17] S. Das and P. Suganthan. [Differential evolution: A survey of the state-of-the-art](#). *IEEE transactions on evolutionary computation*, 15(1):4–31, 2010.
- [18] S. Dempe. [A necessary and a sufficient optimality condition for bilevel programming problems](#). *Optimization*, 25(4):341–354, 1992.
- [19] S. Dempe. [Foundations of bilevel programming](#). MA: Springer US, 2002.
- [20] Y. Diouane, V. Kungurtsev, F. Rinaldi, and D. Zeffiro. [Inexact direct-search methods for bilevel optimization problems](#). *Computational Optimization and Applications*, 88(2):469–490, 2024.
- [21] V. Dogan and S. Prestwich. [Bayesian optimization with multi-objective acquisition function for bilevel problems](#). In *Irish Conference on Artificial Intelligence and Cognitive Science*, pages 409–422. Cham: Springer Nature Switzerland, 2022.
- [22] V. Dogan and S. Prestwich. [Bilevel optimization by conditional Bayesian optimization](#). In *International Conference on Machine Learning, Optimization, and Data Science*, pages 243–258. Cham: Springer Nature Switzerland, 2023.
- [23] E. Dolan and J. Moré. [Benchmarking optimization software with performance profiles](#). *Mathematical Programming*, 91(2):201–213, 2002.
- [24] K. Dzahini, F. Rinaldi, C. Royer, and D. Zeffiro. [Direct-search methods in the year 2025: Theoretical guarantees and algorithmic paradigms](#). *EURO Journal on Computational Optimization*, 13:100110, 2025.
- [25] K. Eggleston and W. Yip. [Hospital competition under regulated prices: application to urban health sector reforms in China](#). *International Journal of health care finance and economics*, 4(4):343–368, 2004.
- [26] M. Ehrhardt and L. Roberts. [Inexact derivative-free optimization for bilevel learning](#). *Journal of mathematical imaging and vision*, 63(5):580–600, 2021.

- [27] O. Ekmekcioglu, N. Aydin, and J. Branke. [Bayesian Optimization of Bilevel Problems](#). Technical Report 2412.18518, arXiv, 2024.
- [28] E. Fermi and N. Metropolis. Numerical solution of a minimum problem. Los Alamos Unclassified Report LA-1492, Los Alamos National Laboratory, Los Alamos, USA, 1952.
- [29] L. Franceschi, P. Frasconi, S. Salzo, R. Grazzi, and M. Pontil. [Bilevel Programming for Hyperparameter Optimization and Meta-Learning](#). In International Conference on Machine Learning, pages 1568–1577, 2018.
- [30] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. [Generative adversarial nets](#). Advances in neural information processing systems, 27, 2014.
- [31] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. C. Courville. [Improved training of wasserstein gans](#). Advances in neural information processing systems, 30, 2017.
- [32] P. Hansen, B. Jaumard, and G. Savard. [New Branch-and-Bound Rules for Linear Bilevel Programming](#). SIAM Journal on Scientific and Statistical Computing, 13(5):1194–1217, 1992.
- [33] R. Hooke and T. Jeeves. [“Direct Search” Solution of Numerical and Statistical Problems](#). Journal of the Association for Computing Machinery, 8(2):212–229, 1961.
- [34] M. Islam, H. Singh, and T. Ray. [A surrogate assisted approach for single-objective bilevel optimization](#). IEEE Transactions on Evolutionary Computation, 21(5):681–696, 2017.
- [35] M. Islam, H. Singh, and T. Ray. [Efficient global optimization for solving computationally expensive bilevel optimization problems](#). In 2018 IEEE congress on evolutionary computation (CEC), pages 1–8, 2018.
- [36] H. Jiang, K. Chou, Y. Tian, X. Zhang, and Y. Jin. [Efficient surrogate modeling method for evolutionary algorithm to solve bilevel optimization problems](#). IEEE transactions on cybernetics, 54(7):4335–4347, 2023.
- [37] D. Jones, M. Schonlau, and W. Welch. [Efficient Global Optimization of Expensive Black Box Functions](#). Journal of Global Optimization, 13(4):455–492, 1998.
- [38] E. Kieffer, G. Danoy, P. Bouvry, and A. Nagih. [Bayesian optimization approach of general bi-level problems](#). In Proceedings of the genetic and evolutionary computation conference companion, pages 1614–1621, 2017.
- [39] T. Kleinert, M. Labbé, I. Ljubić, and M. Schmidt. [A survey on mixed-integer programming techniques in bilevel optimization](#). EURO Journal on Computational Optimization, 9:100007, 2021.
- [40] D. Kraft. A software package for sequential quadratic programming. DFVLR-FB 88–28, Forschungsbericht- Deutsche Forschungs- und Versuchsanstalt für Luft- und Raumfahrt, Braunschweig, Germany, 1988.
- [41] L. LeBlanc and D. Boyce. [A bilevel programming algorithm for exact solution of the network design problem with user-optimal flows](#). Transportation Research Part B: Methodological, 20(3):259–265, 1986.
- [42] R. Lewis, V. Torczon, and M. Trosset. [Direct Search Methods: Then and Now](#). Journal of Computational and Applied Mathematics, 124(1–2):191–207, 2000.
- [43] Z. Liu, Y. Wang, S. Vaidya, F. Ruele, J. Halverson, M. Soljačić, T. Hou, and M. Tegmark. [Kan: Kolmogorov-arnold networks](#). Technical Report 2404.19756, arXiv, 2024.
- [44] K. Ma, L. Rios, H. Zheng, N. Sahinidis, and S. Rajagopalan. [Model-and-search: a derivative-free local optimization algorithm](#). Computational Optimization and Applications, 92(3):889–921, 2025.
- [45] Z. Ma, Z. Huang, J. Chen, Z. Cao, and Y. Gong. [Surrogate learning in meta-black-box optimization: A preliminary study](#). In Proceedings of the Genetic and Evolutionary Computation Conference, pages 1137–1145, 2025.
- [46] A. Madry, A. Makelov, L. Schmidt, D. Tsipras, and A. Vladu. [Towards deep learning models resistant to adversarial attacks](#). Technical Report 1706.06083, arXiv, 2017.
- [47] C. Maheshwari, S. Sasty, L. Ratliff, and E. Mazumdar. [Convergent first-order methods for bi-level optimization and stackelberg games](#). Technical Report 2302.01421, arXiv, 2023.
- [48] T. Man, X. Li, Z. Liu, H. Zhang, B. Gu, and Y. Chang. [Enhancing Black-Box Adversarial Attacks on Discrete Sequential Data via Bilevel Bayesian Optimization in Hybrid Spaces](#). In Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V. 1, pages 1032–1043, 2025.
- [49] A. Mersha and S. Dempe. [Direct search algorithm for bilevel programming problems](#). Computational Optimization and Applications, 49(1):1–15, 2011.
- [50] J. Moré and S. Wild. [Benchmarking Derivative-Free Optimization Algorithms](#). SIAM Journal on Optimization, 20(1):172–191, 2009.
- [51] V. Nguyen, R. Johnson, F. Sup, and B. Umberger. [Bilevel optimization for cost function determination in dynamic simulation of human gait](#). IEEE Transactions on Neural Systems and Rehabilitation Engineering, 27(7):1426–1435, 2019.
- [52] J. Outrata. [On the numerical solution of a class of Stackelberg problems](#). Zeitschrift für Operations Research, 34(4):255–277, 1990.

- [53] G. Papavassilopoulos. [Algorithms for static Stackelberg games with linear costs and polyhedra constraints](#). In 1982 21st IEEE Conference on Decision and Control, pages 647–652, 1982.
- [54] A. Sinha, P. Malo, and K. Deb. [Test problem construction for single-objective bilevel optimization](#). *Evolutionary computation*, 22(3):439–477, 2014.
- [55] M. Staudigl, S. Weissmann, and T. Van Leeuwen. [Derivative-free stochastic bilevel optimization for inverse problems](#). *Computational Optimization and Applications*, 93(3):967–1021, 2025.
- [56] H. Von Stackelberg, A. Peacock, E. Schneider, and T. Hutchison. [The theory of the market economy](#). *Economica*, 20(80):384, 1953.
- [57] A. Wächter and L. T. Biegler. [On the implementation of a primal-dual interior point filter line search algorithm for large-scale nonlinear programming](#). *Mathematical Programming*, 106(1):25–57, 2006.
- [58] Y. Yuan. [A trust region algorithm for Nash equilibrium problems](#). *Pacific Journal of Optimization*, 7(1):125–138, 2011.
- [59] A. Zemkoho and S. Zhou. [Theoretical and numerical comparison of the Karush–Kuhn–Tucker and value function reformulations in bilevel optimization](#). *Computational Optimization and Applications*, 78(2):625–674, 2021.
- [60] D. Zhang and G. Lin. [Bilevel direct search method for leader–follower problems and application in health insurance](#). *Computers and Operations Research*, 41:359–373, 2014.
- [61] S. Zhou, A. Zemkoho, and A. Tin. [BOLIB: bilevel optimization LIBrary of test problems](#). *Bilevel Optimization: Advances and Next Challenges*, 161:563–580, 2020.