Les Cahiers du GERAD ISSN: 0711–2440

Benchmarking derivative-free optimization solvers for CSP plant design using the solar simulator

X. Lebeuf, S. Le Digabel, P. Brillon, M. Diago, C. Audet, C. Tribes

G–2025–70

October 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 derivative-free optimization solvers for CSP plant design using the solar simulator

Xavier Lebeuf [b, c]

Sébastien Le Digabel [b, c]

Philippe Brillon [b]

Miguel Diago [a]

Charles Audet [b, c]

Christophe Tribes [b, c]

[a] *Centre de recherche d'Hydro-Québec, Varennes (Qc), Canada, J3X 1S1*

[b] *Polyterchnique Montréal, Montréal (Qc), Canada, H3T 0A3*

[c] *GERAD, Montréal (Qc), Canada, H3T 1J4*

xavier.lebeuf@polymtl.ca
Sebastien.Le.Digabel@gerad.ca
philippe.brillon@polymtl.ca
diagomartinez.miguel@hydroquebec.com
christophe.tribes@gerad.ca
Charles.Audet@gerad.ca

October 2025
Les Cahiers du GERAD
G–2025–70

Abstract : This work presents a case study where four well-known derivative-free solvers are benchmarked on several instances based on the solar suite of problems: an easy to use, freely available and open-source Concentrated Solar Power (CSP) plant simulator. The solar simulator provides a collection of ten test problems designed to reflect real-world blackbox optimization challenges. The design of CSP plants involves solving complex and often computationally expensive optimization problems. These problems rely on highly diverse models which may be difficult to interface with optimization solvers. As a result, standardized benchmark problems for optimization solvers in the CSP design field are scarce. Such benchmarks are essential for selecting the most appropriate solver for a given optimization problem. The solar problems model numerous components of the power plant, ranging from the design of the heliostat field to the choice of turbine.

Keywords: Blackbox optimization (BBO); Derivative-free optimization (DFO); benchmarking; Concentrated solar power (CSP); solar thermal power

Résumé : Ce travail présente une étude de cas dans laquelle quatre solveurs sans dérivées reconnus sont comparés sur plusieurs instances issues de la suite de problèmes solar : un simulateur de centrale solaire à concentration libre d'accès, à code source ouvert et facile à utiliser. Le simulateur solar propose un ensemble de dix problèmes tests conçus pour refléter les défis réels de l'optimisation de boîte noire. La conception d'une centrale solaire thermodynamique implique la résolution de problèmes d'optimisation complexes et souvent coûteux en calcul. Ces problèmes reposent sur des modèles très divers, qui peuvent être difficiles à interfacer avec des solveurs d'optimisation. Par conséquent, les problèmes de référence standardisés pour les solveurs d'optimisation dans le domaine de la conception de centrales CSP sont rares. De tels benchmarks sont essentiels pour sélectionner le solveur le plus approprié à un problème d'optimisation donné. Les problèmes solar modélisent de nombreux composants de la centrale, allant de la conception du champ d'héliostats au choix de la turbine.

Mots clés : Optimisation de boîte noire; optimisation sans dérivées; analyse comparative; puissance solaire à concentration; puissance solaire thermodynamique

1 Introduction

This work studies simulation-based optimization in the context of Concentrated Solar Power (CSP) plant design. Indeed, such design often involves optimization problems where physical processes are modelled through complex simulations. These practices fall under the category of Blackbox Optimization (BBO) [5]. BBO is a branch of applied mathematics that has been successfully applied in various fields of engineering including physics, chemistry, bioscience and environmental sciences [1].

Formally, an optimization problem consists of minimizing an objective function $f(x) : X \to \overline{\mathbb{R}}$ under constraints $c(x) : X \to \overline{\mathbb{R}}^m$, where $X \subseteq \mathbb{R}^n$ defines bound constraints and $\overline{\mathbb{R}} = \mathbb{R} \cup \{\infty\}$. The feasible region is given by $\{x \in X \,|\, c(x) \leq 0\}$. Allowing f and c to take infinite values conveniently models situations where the evaluation fails to produce valid output. BBO is the field of mathematical optimization where the objective or constraint functions are not explicitly known. Rather, these functions are evaluated via queries to a blackbox for some design parameters $x \in X$. In this work, the blackbox is a numerical code simulating a part of or an entire CSP plant. As such, these evaluation results are the only accessible information to a solver during an optimization.

An optimization solver evaluates the blackbox repeatedly, and each trial point is carefully chosen based on information from previous evaluations. This process continues until a stopping criterion is met, either based on convergence or a predefined evaluation budget, after which the solution $x \in X$ that best minimizes $f(x)$ while all constraints $c(x)$ are satisfied is returned. Solvers may require an initial point x^0. This process is schematized in Figure 1.

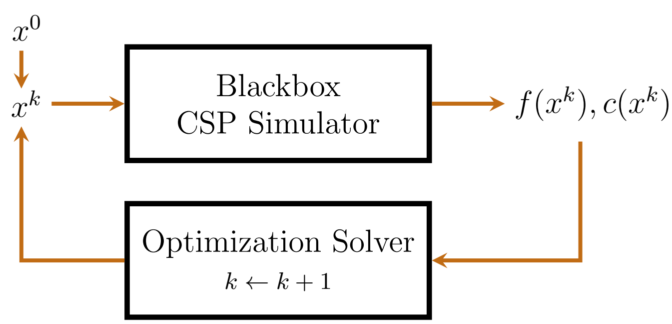


Figure 1: Optimization process in the context of BBO with $k \in \mathbb{N}$ as the iteration counter.

1.1 Motivation

BBO problems are characterized by many challenges, the most common being the following. Each evaluation is expensive, either in computational resources or in time; derivatives are unavailable, or may simply not exist; the blackbox functions contain numerous local optima in addition to being subject to random noise; the simulation might crash during an evaluation. Derivative-free optimization (DFO) solvers are specifically designed to address these challenges and perform effectively under such conditions. However, real engineering problems that exhibit these challenges usually rely on proprietary codes, on highly diverse models that may be difficult to interface with optimization solvers, or are simply not free to use. As a result, DFO solvers are too often benchmarked on problems that do not fully capture the challenges of the field. To address this gap, the solar suite of benchmark problems was introduced in [2, 13] to provide realistic problems to the scientific community. Unlike traditional optimization benchmarks, which often oversimplify engineering problems, solar retains key physical complexities. In this way, the challenges listed above are faithfully reflected in the solar suite.

This study pursues two main objectives. First, it showcases how solar problems can be used by a CSP designer (or more abstractly, by anyone who handles blackbox problems) to compare solvers with the goal of selecting one for a particular purpose. Second, it presents the first large-scale comparison of four well-known blackbox optimization solvers using solar benchmark problems. These solvers are NOMAD [7], HEXALY [10], PRIMA [16] and CMA-ES [11].

The paper is divided as follows. Section 2 describes the solar suite and illustrates its versatility. Section 3 benchmarks the four DFO solvers on some test problems from solar.

2 The SOLAR suite

This section details the content of the CSP plant simulator in Section 2.1, the suite of ten problems is described in Section 2.2, and some indications on how one could use solar to produce results such as the ones in Section 3 are given in Section 2.3.

2.1 Description of the simulator

The simulator involves five components, each controlled by a set of design variables that from the input vector $x \in X$, as well as three auxiliary models. Figure 2 illustrates these components with their corresponding variables. The plant can be partially run, resulting in varying numbers of design variables.

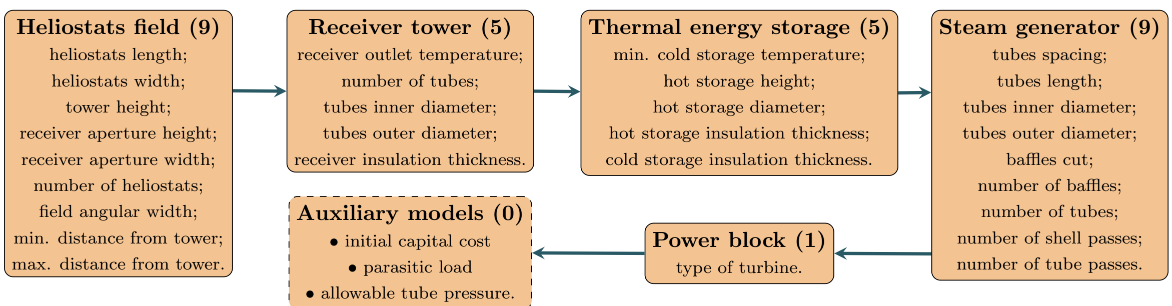
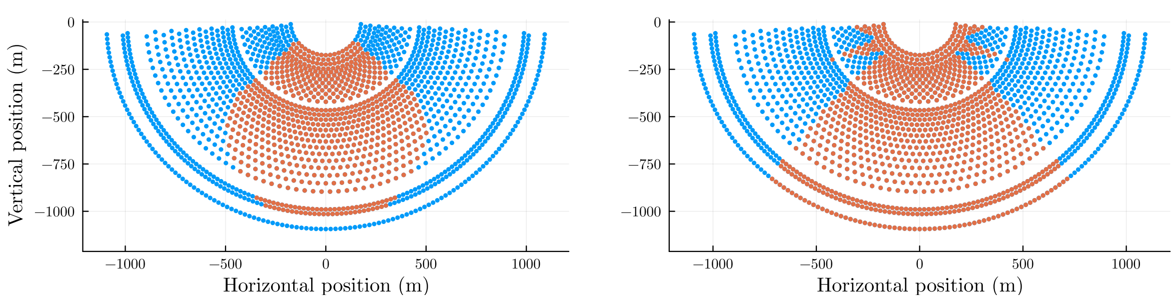


Figure 2: Sequence of models called in the solar simulator if the whole plant is run. The number of design variables of each model is indicated in parentheses.

The **heliostat field** model generates a radially staggered heliostat field parametrized with the heliostat dimensions and the receiver tower height. Subsequently, the positions of individual heliostats are selected based on the other six, in order to maximize optical efficiency as evaluated through ray-tracing Monte Carlo simulations that account for spillage, attenuation, and reflectivity losses. In this model, the number of heliostats is a discrete variable, while the others are continuous. Figure 3, shows two illustrative examples of field generation and position selection by varying two design variables while the other seven are fixed.



(a) $x_5 = 11$, $x_6 = 750$ **(b)** $x_5 = 3$, $x_6 = 950$

Figure 3: Examples of positions selected (in orange) for the same generated field (in blue) and different receiver aperture widths x_5 (in meters) and numbers of heliostats x_6.

The **receiver tower** is modelled as a cavity with a rectangular aperture. Heat lost back to the environment by reflection, re-radiation, convection, and conduction through the receiver wall is also modelled. In this model, the number of tubes is a discrete variable, and the others are continuous.

The **thermal energy** storage system consists of two insulated cylindrical tanks with six continuous design parameters. Heat losses through floor and wall conduction as well as top-surface radiation are modelled based on salt temperature and level. Internal convection effects are neglected, while external radiation and convection to the atmosphere are accounted for under constant wind and normal atmospheric conditions.

The **steam generation system** (or heat exchanger) is modelled as a molten salt–to–water shell-and-tube heat exchanger using the effectiveness–NTU method. In this model, the number of baffles, tubes, shell passes and tube passes are discrete, while the other variables are continuous. This heat exchanger, the receiver, and both storage tanks form the molten salt cycle, where the temperature and flow in each subsystem are interdependent.

The **power block** model depends on a single categorical design variable to decide the type of turbine used. After a turbine is selected, all relevant parameters are retrieved from technical datasets. This includes steam inlet pressures and temperatures needed to operate each turbine.

Finally, the **auxiliary models** are used to add constraints to the optimization problems.

2.2 Description of the ten problems

The model described above has allowed to define ten benchmark problems that involve different subsets of optimization variables. Each problem involves unique objective and constraint functions. Table 1 summarizes the 10 problems of the suite with their main features. Constraints are classified either as a priori or simulated. A priori constraints have an explicit formulation, meaning that they are easy to verify and the CSP simulator is only invoked when they are satisfied [12]. As an example, in solar7, constraint $x_6 \leq x_7$ ensures the inner diameter of the receiver tubes is smaller than their outer diameter, and constraint $x_4 x_7 \leq \frac{\pi}{2} x_2$ ensures that the receiver tubes (outer diameter times number of tubes) fit inside the receiver (aperture width time $\frac{\pi}{2}$). All other constraints in solar7 depend on simulated values.

Table 1: Suite of ten benchmark problems with their description and main features.

Problem	Optimization Problem	# of variables		# of constraints		# of stoch.
		cont.	discr.	simu.	a priori	outputs
solar1	Maximize heliostat field energy output over a 24 h budget with a given budget.	8	1	2	3	1
solar2	Minimize heliostat field surface and satisfy a 24 h energy demand profile.	12	2	7	5	4
solar3	Minimize investment cost and satisfy a 24 h energy demand profile.	17	3	8	5	5
solar4	Similar to solar3 (minimize investment cost), using with more complex models and a different demand profile.	22	7	9	7	6
solar5	Maximize satisfaction of a 30-day demand profile under varying weather conditions.	14	6	8	4	0
solar6	Minimize cost of storage while maintaining a 100 MW output during 24 h.	5	0	6	0	0
solar7	Maximize receiver efficiency under physical constraints.	6	1	4	2	3
solar8	Maximize heliostat performance and minimize investment cost.	11	2	4	5	4
solar9	Maximize power production and minimize losses with a given budget.	22	7	10	7	6
solar10	Similar to solar6, with bound constraints only and penalties added to the objective function.	5	0	0	0	0

For stochastic problems, a replication parameter is available to repeat an evaluation and average the outputs. This parameter can be set to a value in $(0, 1)$ to perform replications until the Gauss-distribution criteria described in [9] is met. In this mode, the given value represents the probability that all stochastic outputs are stabilized within three significant digits. A multi-fidelity parameter is also available for stochastic problems. Fidelity is a value in $[0, 1]$ where the lower the value, the easier convergence criteria in internal numerical methods are reached. This results in faster evaluations for less accurate outputs. The default number of replications is 1 and the default fidelity value is 1.0. See [2] for a detailed description of the problems and their parameters.

2.3 Application examples

Below are some examples of how someone can select a solar problem to conduct benchmarks with the goal of finding the best solver for a specific application.

- The solver must be able to handle simulation failures and a narrow feasible region. Problem solar3 with fidelity 0.5 is chosen because the simulation sometimes crashes at reduced fidelity, and the high number of constraints make feasible points scarce.
- The solver must handle two objective functions: maximize the electrical power production and minimize parasitic losses. Problem solar8 has these exact objective functions for a simulation of the heliostat field and the receiver tower. Problem solar9 is also biobjective, and runs the entire CSP plant.
- The solver only handles continuous variables. Problem solar6 is chosen if a constrained problem is needed, solar10 is for an unconstrained problem.
- The solver must solve a ray-tracing Monte Carlo simulation-based problem. Problem solar1 is selected since it simulates only the heliostat field component and its stochasticity is due to a Monte Carlo simulation of solar ray trajectories.

Once a solar problem (or multiple problems) is chosen, benchmarks are conducted by creating a set of problem instances $\mathcal{P}$. Given a set of solvers $\mathcal{S}$ and a set of solar optimization problems, the benchmarking procedure produces a collection of initial points for each optimization problem. Each pair of optimization problem and initial point forms a problem instance in $\mathcal{P}$. One may also create instances by selecting random seeds for the solvers or the optimization problems if applicable. To find a collection of initial points, the authors recommend using Latin Hypercube Sampling (LHS) [14], a method of sampling a design space with uncorrelated points. Then, the benchmarking procedure launches each solver in $\mathcal{S}$ on each problem instance in $\mathcal{P}$.

All the solvers selected for this study provide a Python interface, which makes this procedure easier. The solar suite is platform-independent and relies only on a random seed value, which may be given as a parameter to solar.

3 Case study

This section compares four DFO solvers on a broad set of problems from the solar suite v1.0 available at `https://github.com/bbopt/solar`. These solvers were selected because they handle inequality-constrained blackbox problems, accept both feasible and infeasible initial points, and make no smoothness assumptions on the problem functions.

- NOMAD [7] is an implementation of the Mesh Adaptive Direct Search (MADS) algorithm [3]. This algorithm only requires an ordering of the evaluated function values to work. No models or gradient approximations are needed, though these tools might be used to help. Consequently, it is best suited for highly discontinuous and unpredictable problems where modelling might be too difficult. Version 4.5.0 is used, available at `https://www.gerad.ca/en/software/nomad`;

- HEXALY [10] is a commercial blackbox optimization solver based on derivative-free trust-region methods. It builds and maintains surrogate models of the objective and constraints using radial basis functions, which are refined as evaluations are collected. This makes it particularly efficient on smooth problems where a good local model can be constructed. Version 14.0 is used, available at https://www.hexaly.com/;
- PRIMA [16] is a collection of model-based derivative-free solvers implementing Powell's algorithms. These methods iteratively build quadratic models of the objective and constraints over a simplex or trust-region. They are most effective on smooth problems where derivatives are unavailable but the functions behave regularly. COBYLA from version 0.7.3 is chosen for this study, available at https://github.com/libprima/prima;
- CMA-ES [11] is an evolutionary algorithm that samples candidate solutions from a multivariate Gaussian distribution, which is iteratively updated using covariance matrix adaptation to better fit the objective and constraint functions. It is well suited for non-convex, noisy, and ill-conditioned problems where global exploration is needed. Version 4.2.0 is used, available at https://github.com/cma-es.

Section 3.1 details the benchmarking methodology. Section 3.2 presents benchmarks on solar10, an unconstrained problem; solar7, a problem with a low number of local minima relative the other solar problems; and solar2, solar3 and solar4, three problems of increasingly high dimension and where feasible points are increasingly scarce.

3.1 Benchmarking methodology

Each variable in the solar suite has a lower bound, and all but a few have an upper bound. Since, HEXALY, PRIMA and CMA-ES require bound constraints for all variables, the value 10^5 is chosen as an artificially high upper bound. Some solar problems have linear constraints that are explicitly known. These constraints are specified to HEXALY and PRIMA, because these solvers have features to handle them directly. Otherwise, the default parameters are selected for all solvers. Across all four solvers, each optimization is conducted with a budget of 100 n_p evaluations, where n_p is the number of variables in problem instance $p \in P$.

Data profiles are used [15] to compare the performance of solvers on a large number of optimizations. A plot of data profiles displays, for each solver, the proportion of τ-*solved* instances as the optimizations progressed. Solver $s \in \mathcal{S}$ is said to have τ-*solved* problem instance $p \in \mathcal{P}$ with initial point x_p^0 after evaluation number $N \in \mathbb{N}$ if

$$\frac{f^0 - f^N}{f^0 - f^*} \geq 1 - \tau,$$

where $f^0 = f(x_p^0)$ if x_p^0 is feasible, f^N is the best feasible objective function value found by solver s on the instance problem up to N evaluations, f^* is the best feasible solution found among all solvers on the instance problem and $\tau \in (0,1)$ is a tolerance. Section 3.2 describes the case where x_p^0 is infeasible.

Define $t_{p,s}$ the smallest number $N \in \mathbb{N}$ for which solver $s \in \mathcal{S}$ has τ-*solved* problem instance $p \in \mathcal{P}$. This number expresses the first evaluation number for a solver where the best improvement from the initial solution is at least $1 - \tau$ times the best improvement among all solvers. Higher τ values make the convergence test harder to satisfy.

Then, for each solver, the plotted value is

$$\text{Portion of } \tau\text{-}\textit{solved} \text{ instances}(\kappa) := \frac{1}{|\mathcal{P}|} \left| \left\{ p \in \mathcal{P} : \frac{t_{p,s}}{n_p + 1} \leq \kappa \right\} \right|,$$

where κ is a number of groups of $n + 1$ evaluations. Such groups are used to allow problems in $\mathcal{P}$ with different dimensions n_p to be compared. It is recommended that at lest 20 instances are used for a data profile. To give a visual intuition, Figure 4 illustrates how solving instances relates to a data profile. The instances from this figure are picked from those of Figure 7. The other solvers are considered to compute f^* values but they are not shown.

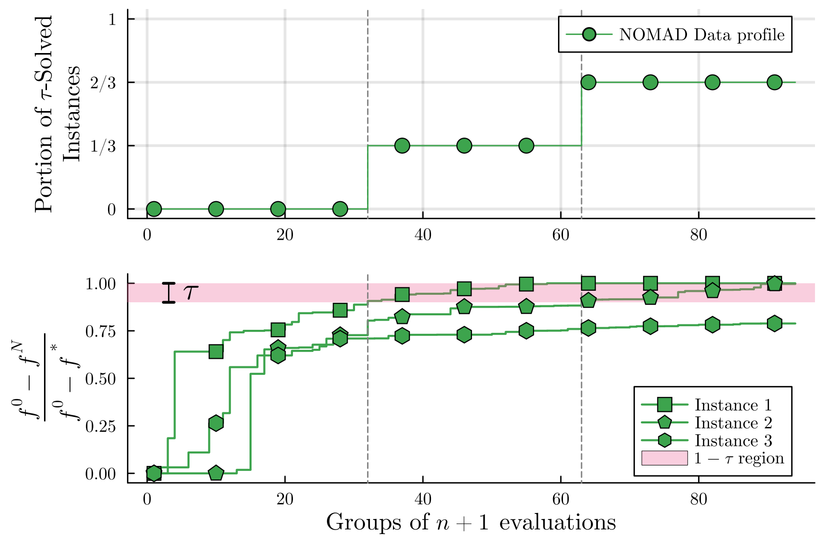


Figure 4: Horizontally aligned plots showing a data profile above with three instances and $\tau = 0.1$, and the convergence graphs of these instances below. Each time a NOMAD optimization comes within $1 - \tau$ of the best improvement corresponds to a bump on the data profile and is marked with a dashed line. Instance 3 is never $\tau-$solved.

See [6, 8] for comprehensive reviews of benchmarking practices and their extensions in the context of BBO.

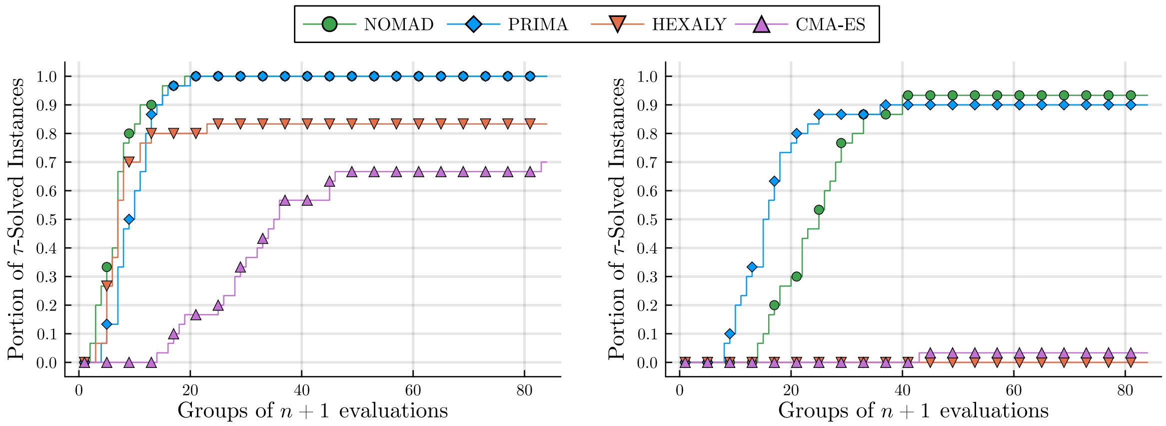
3.2 Numerical results

The procedure described in Section 2.3 is followed. To find numerous initial points and construct just as many problem instances on which the solvers are compared, a LHS method with 10^4 points is performed for each of the five selected solar problems. Table 2 shows the feasibility of the sampled points. Either none or a single feasible point is generated for each of the first three instances and the a priori constraints are difficult to satisfy for solar4. For solar7, approximately half of the points do not satisfy the a priori constraints, and another third are infeasible. Finally, all points are feasible for solar10.

Table 2: Number of points that i- do not satisfy a priori constraints; ii- satisfy a priori constraints but are infeasible; iii- are feasible.

Problem	Number of points infeasible a priori	Number of points infeasible, feasible a priori	Number of feasible points
solar2	8147	1852	1
solar3	9098	901	1
solar4	9990	10	0
solar7	5392	3254	1354
solar10	0	0	10^4

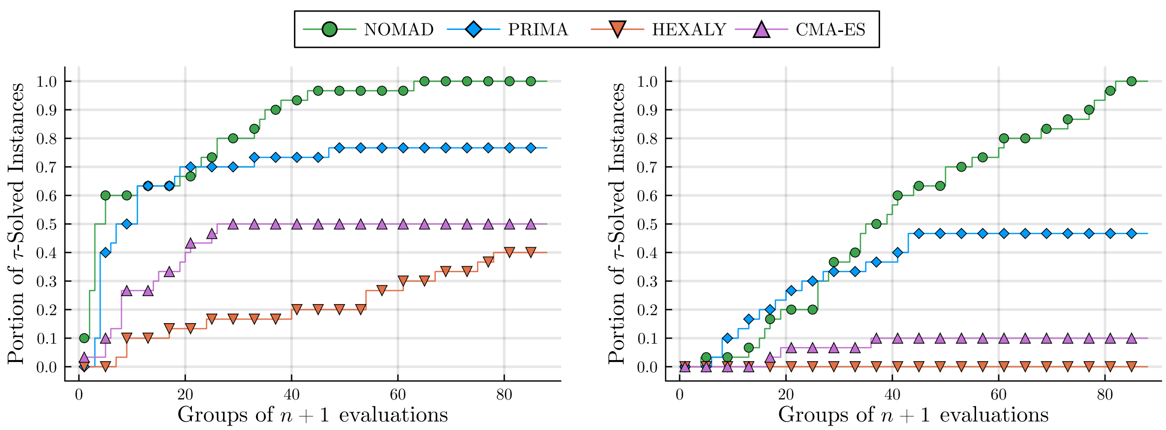
Since feasible points are easy to find for solar7 and solar10 is unconstrained, only feasible points are selected for these problems. Table 2 indicates that the other three problems are significantly more challenging, as they probably have a narrow feasible domain. To this effect, they are tested on feasible initial points, and then solar3 and solar4 are tested on infeasible initial points. From the LHS conducted on solar10, 30 initial points are selected to form 30 problem instances. Figure 5 displays the results for two different values of τ.



(a) $\tau = 10^{-2}$ **(b)** $\tau = 10^{-4}$

Figure 5: Data profiles on problem solar10 with 30 different feasible initial points.

From the LHS conducted on solar7, 30 out of the 1354 feasible points are chosen as initial points to form 30 problem instances. Figure 6 displays the results for two different values of τ.

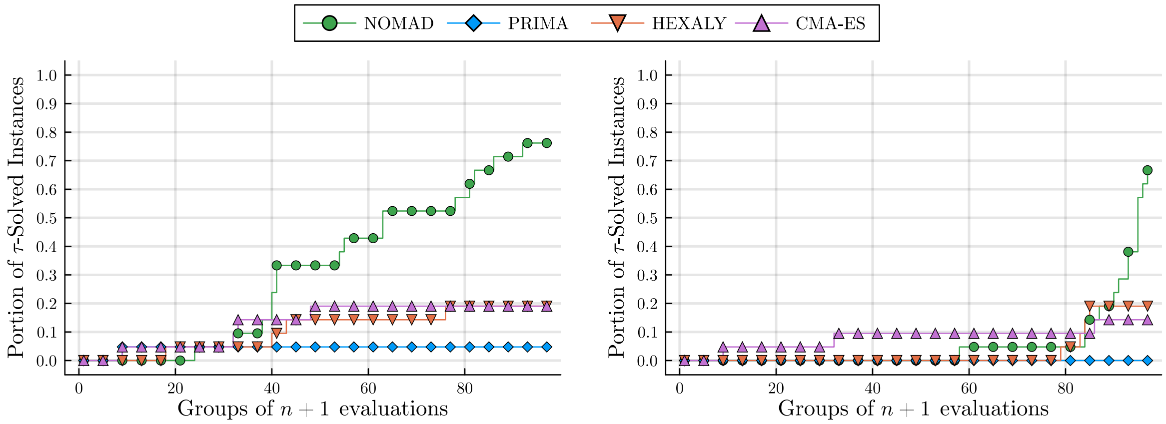


(a) $\tau = 0.1$ **(b)** $\tau = 10^{-2}$

Figure 6: Data profiles on problem solar7 with 30 different feasible initial points.

The LHS methods conducted on solar2, solar3 and solar4 found only two feasible points across all three problems. As such, these two points along with 19 other feasible points previously known by the authors are selected. The problem instances are the following: five initial points for solar4, eight initial points for solar3 and eight initial points for solar2. Figure 7 displays the results for two different values of τ.

The a priori constraints are checked by the solar code, and the CSP simulation is not run when one is violated. These constraints are the easiest to compute due to their analytical nature and a gradient-based method could be used to satisfy them. Starting in a region where the simulator is executed yields more significant BBO benchmarks. To this effect, infeasible points that satisfy a priori constraints are selected. All ten solar4 infeasible points from Table 2 that satisfy a priori constraints, and 20 of the 901 points for solar3 are selected to form 30 problem instances. The conducted optimizations reveal that solar3 and solar4 are indeed difficult problems, as the solvers spend most of their efforts on finding a feasible point rather than optimizing the objective function value. Table 3 quantifies this observation. Only the NOMAD solver finds feasible points for both solar problems.



(a) $\tau = 0.1$

(b) $\tau = 10^{-3}$

Figure 7: Data profiles on problems solar2, solar3 and solar4 with 21 feasible initial points.

Table 3: Number of problem instances out of 30 where a feasible point is found for each solver.

Problem	Number of instances	Number of instances where a feasible point is found			
		NOMAD	PRIMA	HEXALY	CMA-ES
solar3	20	3	0	1	5
solar4	10	4	0	0	0

The data profiles are not informative since the portion of τ-solved instances is very low. The alternative is to compare the solvers on their ability to approach a feasible solution. Consider the constraint violation function $h : \mathbb{R}^m \to \overline{\mathbb{R}}$ introduced in [4]:

$$h(x) := \begin{cases} \sum_{j=1}^{m} (\max\{c_j(x), 0\})^2 & \text{if } x \in X \\ \infty & \text{otherwise.} \end{cases}$$

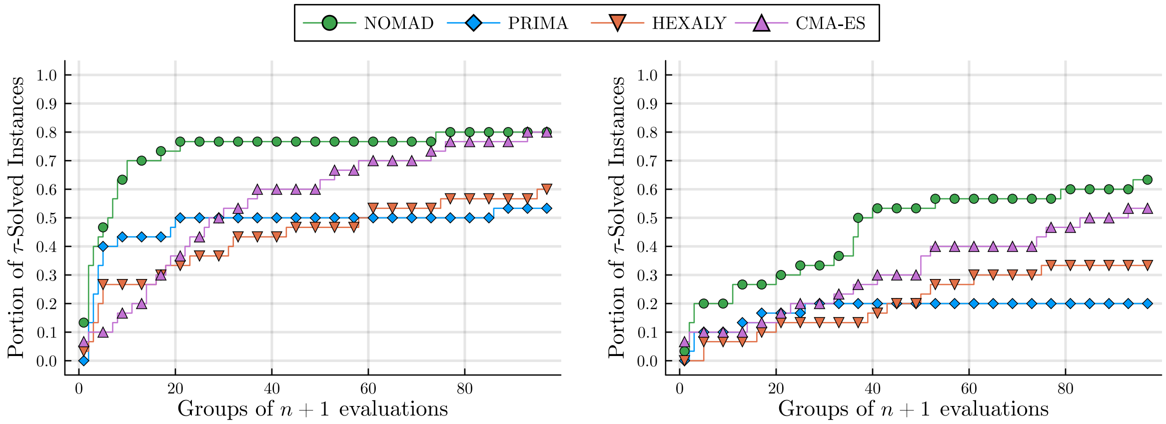
This function measures the degree to which constraints are violated at a given point, and $h(x) = 0$ if and only if x is feasible. In Figure 8, for each problem instance $p \in \mathcal{P}$, $f^0 = h(x_p^0)$, f^* is the best h value found among all solver $s \in \mathcal{S}$, and f^N is the best value of h found by a solver up to evaluation number $N \in \mathbb{N}$. The figure suggests that NOMAD outperforms the other methods for approaching the feasible region.

To better solve these hard problems, one might use adapted parameters for a solver and use a larger blackbox evaluations budget, but this exceeds the scope of the present work.

4 Discussion

This paper highlights solar, the first freely available and open source suite of benchmarking problems specifically designed for BBO. For a CSP plant designer that needs to chose the best solver for a specific application, the solar suite offers 10 problems with varying properties. Once a CSP designer has chosen the solar problems that most closely reflects their challenges, this work offers a methodology to compare a selection of solvers. This extends to anyone who desires to benchmark solvers on BBO problems with specific problem properties that can be found in solar.

For discontinuous, stochastic blackbox optimization problems, *mesh-based* solvers (such as NOMAD) typically perform better than *model-based* solvers (PRIMA, HEXALY and CMA-ES). Conversely, model-based approaches tend to perform better on smooth problems. The case study reveals that for the problems tested, the mesh-based method reliably performs better than the other. This is a sign that solar successfully captures the difficulties encountered with real BBO problems.



(a) $\tau = 0.1$ **(b)** $\tau = 10^{-2}$

Figure 8: Data profiles of $h(x)$ on problems, solar3 and solar4 with infeasible initial points.

Among the model-based solvers, no method outperforms the others in all problems. As a matter of fact, considering the higher precision data profiles in each pair of graph, each model-based solver performs the best in at least one benchmark. This indicates that the different solar problems are well diversified.

We hope that this contribution will encourage the use of realistic benchmark problems when comparing solvers in the field of BBO.

References

[1] S. Alarie, C. Audet, A.E. Gheribi, M. Kokkolaras, and S. Le Digabel. Two decades of blackbox optimization applications. EURO Journal on Computational Optimization, 9:100011, 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. Optimization and Engineering, 26(3):1815–1861, 2025.

[3] C. Audet and J.E. Dennis, Jr. Mesh Adaptive Direct Search Algorithms for Constrained Optimization. SIAM Journal on Optimization, 17(1):188–217, 2006.

[4] C. Audet and J.E. Dennis, Jr. A Progressive Barrier for Derivative-Free Nonlinear Programming. SIAM Journal on Optimization, 20(1):445–472, 2009.

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

[6] C. Audet, W. Hare, and C. Tribes. Benchmarking constrained, multi-objective and surrogate-assisted derivative-free optimization methods. Technical Report G–2025–36, Les cahiers du GERAD, 2025.

[7] 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.

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

[9] M. Bicher, M. Wastian, D. Brunmeir, and N. Popper. Review on Monte Carlo Simulation Stopping Rules: How Many Samples Are Really Enough? Simulation Notes Europe, 32(1):1–8, 2022.

[10] F. Gardi and T. Benoist. Hexaly. Software available at https://www.hexaly.com/.

[11] 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.

[12] 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.

[13] M. Lemyre Garneau. Modelling of a solar thermal power plant for benchmarking blackbox optimization solvers. Master's thesis, Polytechnique Montréal, 2015. Available at https://publications.polymtl.ca/1996/.

[14] M.D. McKay, R.J. Beckman, and W.J. Conover. A comparison of three methods for selecting values of input variables in the analysis of output from a computer code. Technometrics, 21(2):239–245, 1979.

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

[16] Z. Zhang. PRIMA: Reference Implementation for Powell's Methods with Modernization and Amelioration. available at http://www.libprima.net,DOI:10.5281/zenodo.8052654, 2023.