Les Cahiers du GERAD ISSN: 0711–2440

Hierarchically constrained blackbox optimization

S. Alarie, C. Audet, P. Jacquot, S. Le Digabel

G–2021–65

November 2021

Citation suggérée : S. Alarie, C. Audet, P. Jacquot, S. Le Digabel (Novembre 2021). Hierarchically constrained blackbox optimization, Rapport technique, Les Cahiers du GERAD G–2021–65, GERAD, HEC Montréal, Canada.

Avant de citer ce rapport technique, veuillez visiter notre site Web (https://www.gerad.ca/fr/papers/G-2021-65) afin de mettre à jour vos données de référence, s'il a été publié dans une revue scientifique.

Suggested citation: S. Alarie, C. Audet, P. Jacquot, S. Le Digabel (November 2021). Hierarchically constrained blackbox optimization, Technical report, Les Cahiers du GERAD G–2021–65, GERAD, HEC Montréal, Canada.

Before citing this technical report, please visit our website (https://www.gerad.ca/en/papers/G-2021-65) 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, 2021
– Bibliothèque et Archives Canada, 2021

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, 2021
– Library and Archives Canada, 2021

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

Hierarchically constrained blackbox optimization

Stéphane Alarie [a, b]

Charles Audet [b, c]

Paulin Jacquot [b, c]

Sébastien Le Digabel [b, c]

[a] *Hydro-Québec & IREQ, Montréal (Qc), Canada, J3X 1S1*

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

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

alarie.stephane@hydroquebec.com
charles.audet@gerad.ca
jacquot.paulin@gmail.com
Sebastien.le.digabel@gerad.ca

November 2021
Les Cahiers du GERAD
G–2021–65

Abstract : In blackbox optimization, evaluation of the objective and constraint functions is time consuming. In some situations, constraint values may be evaluated independently or sequentially. The present work proposes and compares two strategies to define a hierarchical ordering of the constraints and to interrupt the evaluation process at a trial point when it is detected that it will not improve the current best solution. Numerical experiments are performed on a closed-form test problem.

Keywords: Blackbox optimization, derivative-free optimization, constrained optimization

Acknowledgements: This work has been funded by a MITACS Elevate grant in collaboration with Hydro-Québec.

1 Introduction

Consider the constrained optimization problem

$$(P) \qquad \begin{array}{rl} \min\limits_{x \in \mathcal{X}} & f(x) \\ \text{s.t.} & c_1(x) \le 0 \\ & c_2(x) \le 0 \\ & \vdots \\ & c_m(x) \le 0 \end{array}$$

in which $\mathcal{X}$ represents the domain of the objective $f : \mathcal{X} \to \overline{\mathbb{R}}$ and of the constraint functions $c_j : \mathcal{X} \to \overline{\mathbb{R}}$, for $j \in \{1, 2, \ldots, m\}$. The domain $\mathcal{X}$ is usually $\mathbb{R}^n$, $\mathbb{N}^n$ or some bound-constrained subset of one of these spaces. The entire feasible region is compactly denoted by

$$\Omega \;=\; \{\; x \in \mathcal{X} \mid c_j(x) \le 0 \;\forall j \in \{1, 2, \ldots, m\} \;\}.$$

The present work considers the situation where the functions defining (P) are provided as *blackboxes* [3], usually in the form of computer codes, but more specifically, cases in which each constraint c_j and the objective f may be evaluated independently or sequentially. We adopt the convention that $c_j(x) = +\infty$ if the evaluation of $c_j(x)$ could not be terminated, for example if the simulation process crashed. A similar convention holds for the objective function f.

More precisely, we consider the case where the j-th blackbox returns the value of the j-th constraint c_j for j ranging from 1 to m, and the $(m+1)$-th blackbox returns the objective function f. When studying a trial point $x \in \mathbb{R}^n$, a decision may then be taken after each constraint evaluation to decide whether or not to spend more effort in evaluating the remaining functions.

The objective of this work is to solve Problem (P) while reducing as much as possible the overall computational effort. The motivating problem is the PRIAD project at Hydro-Québec TransÉnergie [9] to optimize the maintenance strategies of its electrical power grid. The optimization model is still being developed, but each evaluation will go through a sequence of five blackboxes given the frequencies of preventive maintenance to be performed on grid equipment: i- feasibility with the workload capacity; ii- resulting equipment failure probabilities; iii- scenario generation of unavailable equipment; iv- power flow simulations to quantify energy delivery interruptions; v- risk and cost assessment of load shedding and corrective maintenance. The first four are related to constraints and the last to the objective function. They are chained so that the outputs of one are inputs of the next. Computational time also increases significantly from blackbox to blackbox. While the first two will take a few seconds to complete, the fourth will require days.

We propose two different two-phase procedures, in which the first phase is dedicated to finding a feasible solution, and the second phase is devoted to the optimization of (P). Both procedures exploit the flexibility of the Mesh Adaptive Direct Search (Mads) algorithm [1] designed for constrained blackbox optimization.

This document is composed of two main sections. Section 2 describes a pair of algorithmic procedures to interrupt evaluations and Section 3 illustrates the procedures on an analytical tension/compression spring design problem. Concluding remarks and potential future work close the paper in Section 4.

2 Interruption of futile blackbox evaluations

Subsections 2.2 and 2.3 describe algorithmic procedures to interrupt the sequence of constraint evaluations. Both procedures rely on the Mads framework described in the next subsection.

2.1 The Mesh Adaptive Direct Search algorithm (Mads)

Mads is a class of direct search algorithms designed to solve Problem (P). It was first proposed in the paper [1], but has undergone many improvements. The latest description is found in the recent paper [4]. The simplest form of this class of algorithms is presented in the textbook [3].

Mads is an iterative algorithm that attempts at each iteration to generate a trial point that would be an improvement over the current best-known solution called the incumbent. Mads is initiated with one trial point $x^0 \in \mathcal{X}$ that is not required to satisfy the constraints c_j. Each trial point $x \in \mathbb{R}^n$ is sent to the blackbox for the evaluation of the constraints $c_j(x)$ and objective $f(x)$ function values. The incumbent solution at iteration k is denoted x^k.

Two mechanisms to handle the constraints are proposed in the blackbox optimization literature. Both use the *constraint violation function*

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

inspired from filter methods [7] and adapted in [2]. The constraint violation function value $h(x)$ is zero if and only if x belongs to the feasible region Ω.

The first mechanism, referred to as the Extreme Barrier (EB) approach, is a two-phase method solving a pair of optimization problems without general constraints. The first phase attempts to find a feasible solution by minimizing the constraint violation function h from the starting point x^0. It is only invoked if x^0 is infeasible with respect to the constraint $c(x) \leq 0$. At the starting point, the constraint violation function value $h(x^0)$ is finite since $x^0 \in \mathcal{X}$. If the first phase fails to generate a feasible point, then Problem (P) is declared infeasible and the process terminates. Otherwise, as soon as a feasible point $\bar{x}$ is generated, the first phase is stopped. The second phase is then launched, and consists in minimizing the extreme barrier function

$$f_\Omega(x) := \begin{cases} f(x) & \text{if } x \in \Omega, \\ +\infty & \text{if } x \notin \Omega \end{cases}$$

from the feasible starting point $\bar{x}$.

The second mechanism to handle the constraints is called the Progressive Barrier (PB) [2], and consists in analyzing the bi-objective problem of minimizing both the objective function f and the constraint violation function h. In order to apply PB, all the constraint functions c_j need to be evaluated at the trial points. We next introduce two novel procedures that allow incomplete evaluations of the constraint functions.

2.2 The interruptible extreme barrier procedure

A first approach to solve Problem (P) consists in launching Mads with the two-phase extreme barrier, but to simply interrupt the sequence of constraint evaluations as soon as one is violated.

The first phase of the approach, the feasibility phase, is again to minimize the constraint violation function h as for EB above, from a user-provided starting point x^0, but to terminate the evaluations of the constraints as soon as the cumulative constraint violation function exceeds the incumbent value. Indeed, the partial sums

$$s_\ell(x) := \sum_{j=1}^{\ell} \max(c_j(x), 0)^2$$

appearing in h are monotone increasing with respect to the index ℓ and therefore, the incumbent solution x^k is replaced by a trial point $t \in \mathcal{X}$ if and only if

$$s_\ell(t) < h(x^k) \quad \text{for every index} \quad \ell \in \{1, 2, \ldots, m\}.$$

The second phase solves Problem (P) from the feasible starting point $\bar{x}$ produced by the first phase (if it succeeded). Again, the evaluations of the constraints is interrupted as soon as a violation is detected. Algorithm 2.1 gives the pseudo-code of the two-phase interruptible extreme barrier procedure, denoted (Int).

Algorithm 2.1. Two-phase interruptible extreme barrier (Int)

Given the objective function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, constraints $c_j : \mathbb{R}^n \to \overline{\mathbb{R}}$ for $j \in \{1, 2, \ldots, m\}$ and starting point $x^0 \in \mathcal{X}$.

1. Feasibility phase
    - Apply Mads with the extreme barrier from $x^0 \in \mathcal{X}$ to solve $\min_x h(x)$
        - For a trial point $t \in \mathcal{X}$, evaluate $\{c_\ell(t)\}_{\ell=1}^m$ in sequence,
          but reject t and interrupt evaluation as soon as $s_\ell(x) \geq h(x^k)$ for some index ℓ
        - Interrupt Mads and terminate the feasibility phase as soon as a
          feasible point $\bar{x} \in \Omega$ is generated and go to Phase 2
    - If Mads failed to generate a feasible point,
      terminate by concluding that no point in Ω was found
2. Optimization phase
    - Apply Mads with the extreme barrier (EB) starting from $\bar{x} \in \Omega$ to solve Problem (P)
        - For a trial point $t \in \mathcal{X}$, evaluate $\{c_\ell(t)\}_{\ell=1}^m$ in sequence,
          but reject t and interrupt evaluation as soon as $c_\ell(t) > 0$ for some index ℓ

The next result gives sufficient conditions to ensure that Mads with EB and Int behave identically (in the sense that they generate the same sequence of iterates), however, the latter requires a lower computational effort. The main condition is that no dynamic surrogate model is used by the algorithm, because the models are constructed by using previously evaluated function values. Therefore, more points will be available to construct models to the EB procedure than to the Int procedure.

Proposition 1. Mads *with* EB *and* Int *produce the same sequence of iterates when launched on the same Problem* (P) *from the same starting point and without using dynamic surrogate models.*

Proof. If EB and Int start an iteration from the same incumbent solution and with the same algorithmic parameters and random seeds, both will create the same list of trial points when no dynamic models are used.

First, consider feasibility phase. If the starting point x^0 satisfies all the constraints, then both the Int and EB approaches conclude the phase with $\bar{x} = x^0$. Otherwise, both approaches attempt to minimize $h(x)$ on $\mathcal{X}$. At iteration k, any trial point t outside of $\mathcal{X}$ is rejected and x^{k+1} is set to x^k, the current incumbent solution. If t belongs to $\mathcal{X}$, then there are two possibilities:

i- If $h(t) < h(x^k)$, then both approaches will set x^{k+1} to t;

ii- If $h(t) \geq h(x^k)$, then both approaches will set x^{k+1} to x^k.

It follows that the feasibility phase of both approaches behave identically.

Second, consider the optimization phase. Any trial point t outside of Ω is rejected by both approaches and x^{k+1} is set to x^k. If t belongs to Ω, then there are again two possibilities:

i- If $f(t) < f(x^k)$, then both approaches will set x^{k+1} to t;

ii- If $f(t) \geq f(x^k)$, then both approaches will set x^{k+1} to x^k.

Once more, it follows that both optimization phases behave identically. □

2.3 The hierarchical satisfiability extreme barrier procedure

A second approach to solve Problem (P) while reducing the computational burden sequentially solves a collection of problems parameterized by the constraint index $j \in \{1, 2, \ldots, m\}$, that minimize the function c_j subject to the constraint with lesser indices

$$(P_j) \qquad \begin{array}{rl} \min\limits_{x \in \mathcal{X}} & c_j(x) \\ \text{s.t.} & c_1(x) \le 0 \\ & c_2(x) \le 0 \\ & \vdots \\ & c_{j-1}(x) \le 0 \end{array}$$

The feasibility phase starts by solving the unconstrained Problem (P_0) from the user-provided starting point $x^0 \in \mathcal{X}$. The algorithm then adds one constraint at a time, treats it with the extreme barrier and minimizes the next one by sequentially solving (P_1) to (P_m). Evaluations are interrupted as soon as a constraint is violated, and the entire process is terminated when Mads is unable to find a feasible solution to one of Problems (P_j). The optimization phase is identical to that of the Int approach.

The pseudo-code of the hierarchical satisfyability procedure called (Hier) is presented next.

Algorithm 2.2. Hierarchical satisfiability with the extreme barrier (Hier)

Given the objective function $f : \mathbb{R}^n \to \overline{\mathbb{R}}$, constraints $c_j : \mathbb{R}^n \to \overline{\mathbb{R}}$ for $j \in \{1, 2, \ldots, m\}$ and starting point $x^0 \in \mathcal{X}$.

1. Feasibility phase
   - For each index j varying from 1 to m,
     - Apply Mads with the extreme barrier (EB) from $x^{j-1} \in \mathcal{X}$ to solve Problem (P_j)
       - Terminate Mads as soon as a feasible point is found and denote it x^j
       - For a trial point t, evaluate $\{c_\ell(t)\}_{\ell=1}^{j-1}$ in sequence, but interrupt the evaluation process as soon as $c_\ell(t) > 0$ for some index ℓ
     - Terminate the entire algorithm if no feasible point is found by Mads
2. Optimization phase
   - Apply Mads with EB from the starting point $x^m \in \Omega$ to solve Problem (P)
     - For a trial point t, evaluate $\{c_\ell(t)\}_{\ell=1}^{m}$ in sequence, but interrupt the evaluation process as soon as $c_\ell(t) > 0$ for some index ℓ

The Hier approach does not produce the same sequence of trial points than Mads with EB nor Int.

3 Illustration on a Tension/Compression Spring Design problem

We compare the interruptible and hierarchical approaches described above with the baseline given by direct calls to the NOMAD implementation [10] of the Mads algorithm, using the extreme or progressive barrier to handle constraints. This results in four optimization algorithmic procedures:

- Int: the interruptible extreme barrier approach from Section 2.2;
- Hier: the hierarchical satisfiability extreme barrier approach from Section 2.3;
- EB: Mads with the extreme barrier to handle constraints [1];
- PB: Mads with the progressive barrier to handle constraints [2];

The Tension/Compression Spring Design problem (TCSD) consists of minimizing the weight of a spring under mechanical constraints. The problem was originally introduced in [5] and has also been considered in [8] along with other nonlinear engineering problems. Although it is not a blackbox optimization problem because closed-form expressions of all functions are known, we use it to illustrate the behaviour of the algorithmic procedures.

This problem has three bound-constrained variables and four nonlinear constraints. The design variables define the geometry of the spring. The constraints concern limits on outside diameter (1b), surge frequency (1c), minimum deflection (1d) and shear stress (1e):

$$f(x) = x_1^2\, x_2\, (x_3 + 2) \tag{1a}$$

$$c_1(x) = \frac{1}{1.5}(x_1 + x_2) - 1 \;\le\; 0 \tag{1b}$$

$$c_2(x) = 1 - 140.45\,\frac{x_1}{x_2^2\,x_3} \;\le 0 \tag{1c}$$

$$c_3(x) = 1 - \frac{x_2^3\,x_3}{71785\,x_1^4} \;\le\; 0 \tag{1d}$$

$$c_4(x) = \frac{4\,x_2^2 - x_1\,x_2}{12566(\,x_1^3\,x_2 - x_1^4)} + \frac{1}{5108\,x_1^2} - 1 \;\le\; 0 \tag{1e}$$

This problem is difficult in two ways. It is hard to find feasible solutions and it possesses a large number of local optima.

The best known solution, denoted x^* hereafter, and the bounds on the variables, denoted by ℓ and u, are given in Table 1. The objective function value of x^* is $f(x^*) = 0.0126652$.

Table 1: Bounds and best know feasible point for the TCSD problem.

variable		ℓ	u	x^*
x_1	mean coil diameter	0.05	2.0	0.051686
x_2	wire diameter	0.25	1.3	0.35666
x_3	active coils length	2.0	15.0	11.29231

For illustrative purposes, the *costs* (τ_k) of evaluating the functions is related to the evaluation time and is set to be the *number of multiplications and divisions* for each constraint. Table 2 summarizes these costs. The constraints are ordered so that their costs increase with the index number.

Table 2: Costs of evaluating the functions defining the TCSD problem.

function	c_1	c_2	c_3	c_4	f
cost τ_k	1	4	8	14	3

In the numerical experiments below, the stopping criteria consists of an overall budget of 10,000 multiplications and divisions required by the evaluation of the functions.

The four algorithmic procedures are applied to 40 instances of the TCSD problem by randomly generating infeasible starting points within the bound-constrained domain using an uniform distribution.

Two series of tests are conducted. The first one orders the constraints so that the least expensive are evaluated first and the second one orders the constraints by considering first the ones that are most likely to be violated.

Constraints ordered by increasing evaluation cost

The four constraints are ordered as in Table 2 and results are summarized in Table 3. The left part of the table shows the average cost to reach the first feasible trial point encountered by each procedures. The columns c_1 to c_4 and f indicate the average number of times that each function is called to reach the final objective function value. Even though there are no explicit mechanism to interrupt the sequence of constraint evaluations with the algorithmic procedures EB and PB, the number of evaluations is not constant. For both procedures the table reveals that the objective f was not evaluated as often as the constraints. The reason is that there were trial points for which the constraint c_4 generated a division by zero and this lead to an immediate termination of the simulation without evaluating the objective function f.

Table 3: Average costs for reaching feasibility and final objective function values within an allocated budget of 10,000 multiplications and divisions with constraints ordered by evaluation cost.

	Feasible	Final number of evaluations					Final objective $f(x^*)$	
Proc.	Avg. cost	c_1	c_2	c_3	c_4	f	instances avg.	best instance
EB	1479.9	334.0	334.0	334.0	334.0	333.7	0.0129921	0.0126654
PB	2199.4	334.0	334.0	334.0	334.0	333.8	0.0131401	0.0126656
Int	939.0	457.3	449.4	440.2	264.0	179.1	0.0129580	0.0126659
Hier	833.8	447.8	437.5	429.6	278.5	160.2	0.0133600	0.0126654

The right part of Table 3 shows the final objective function value after that the budget of 10,000 multiplications and division is spent. One column gives the average value and the other gives the lowest objective function value over the 40 instances.

With the Int and Hier procedures, the number of times that the functions are called decreases monotonically from c_1 to f. The table shows that the number evaluations does not vary importantly for the constraints c_1, c_2 and c_3. This is explained by the fact that c_1 and c_2 are strictly satisfied at the optimal solution x^* and consequently the evaluations are not often interrupted for most trial points. The constraints c_3 and c_4 are binding at x^* and thus a significant number of evaluations of c_4 and f are spared. The table also shows that the different procedures require very different costs to generate a first feasible point. On average, Hier finds a first feasible point almost 60% faster than PB.

Figure 1 compares the four algorithmic procedures with data profiles [12] that show the proportion of instances solved with a 5% tolerance (i.e., a 5% gap to the best known value $f(x^*)$) versus the evaluation cost. None of the curves are connected to the ordinate axis because of the infeasible starting points. Each curve starts when the procedure has solved one of the 40 instances within 5%. For Int, Hier and EB algorithmic procedures, the first instance is solved with the same effort, while the PB approach requires more multiplications and divisions. This is coherent with the fact that PB places more effort around infeasible points with promising objective function values [2].

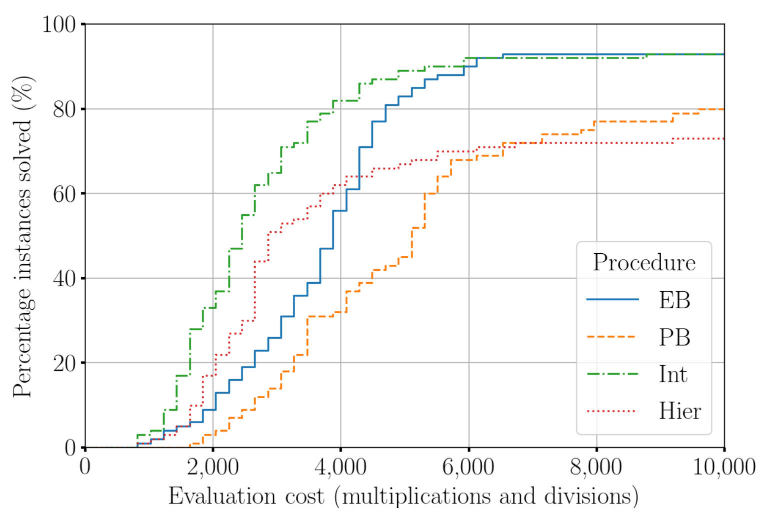

Figure 1: Data profiles showing performance of the different procedures on the TCSD problem, considering constraints in increasing order of evaluation cost (criterion: optimality gap of 5%).

Over most of the duration of the optimization, the curve corresponding to the Int procedure dominates the others. Close inspection of the figure reveals that at an evaluation cost of approximately 7,000, the EB procedure slightly outperforms the Int procedure. This does not contradict Proposition 1 because the construction and utilisation of dynamic quadratic models of the objective and constraint functions was enabled, as this is the default option in NOMAD. This behaviour suggests that the use of quadratic models on this simple TCSD problem helps to accelerate the convergence when the basin containing a local optima is found. When the number of iterations is low, the models are not as useful, and both the Int and Hier procedures benefit from interrupting the calls to the simulation.

Constraints ordered by empirical infeasibility

There are situations where one knows which constraints are difficult to satisfy, which ones are easily satisfied and which ones are not. For example, Rio Tinto's engineers know that maintaining a small flooding risk for the Kemano hydroelectric system is much more challenging than ensuring continuous smelter operations [6]. For the TCSD problem, we simulated that knowledge by evaluating the functions at 5,000 points using Latin Hypercube sampling [11] in the domains delimited by the bounds listed in Table 1. Table 4 gives the proportion of sample points that are feasible for each constraint.

Table 4: Proportion of feasible points for each constraint when sampling 5,000 points, ranked in increasing order.

Constraint	c_3	c_1	c_2	c_4
Feasibility (%)	2	34	99	99

Ordering the constraints so that the most likely to be infeasible are evaluated first will potentially trigger the interruption at a lower computational cost. Results of the performances of the different procedures on the TCSD problem are shown in Table 5.

Table 5: Average costs for reaching feasibility and final objective function values within an allocated budget 10,000 multiplications and divisions, with constraints ordered by empirical infeasibility.

	Feasible	Final number of evaluations					Final objective $f(x^*)$	
Proc.	Avg. cost	c_3	c_1	c_2	c_4	f	instances avg.	best instance
EB	1264.8	334.0	334.0	334.0	334.0	333.8	0.0130931	0.0126653
PB	1934.4	334.0	334.0	334.0	334.0	333.7	0.0131127	0.0126656
Int	863.8	485.4	290.7	290.3	280.2	195.2	0.0130218	0.012668
Hier	863.1	473.1	309.9	309.9	296.5	165.4	0.0133596	0.0126653

A first observation is that results of the EB and PB procedures differ from those listed in Table 3. This discrepancy is due to the random selection of starting points. A second observation is that the average cost for the Int procedure drops by approximately 8% for the generation of a first feasible solution, and increases by 3.5% for Hier. The average and best final objective function values are comparable for all four procedures. Figure 2 shows the corresponding data profiles with a 5% tolerance.

The gain made by the Int and Hier procedures over the EB and PB approaches are even more significant when the cost is low. For example, at an evaluation cost of 4,000, Hier and Int have solved around 80% of the instances, around twice as much as the procedures EB and PB. The Int procedure dominates all others on the totality of the graph, except that the Hier procedure is slightly above at the moment when a first feasible solution is produced.

4 Discussion

We have presented approaches that exploit the flexibility of the Mads algorithm to interrupt the evaluation sequence of the constraints as soon as one is shown to be infeasible. Complex industrial problems, such as the problem PRIAD mentioned in introduction, will undoubtedly benefit from evaluation interruptions, and this without losing solution quality. Numerical experiments suggest that the order in which the constraints are evaluated affects the behaviour. Future work include dynamically ordering the constraints, as well as inserting the evaluation of the objective function in the sequence, and more importantly, applying this approach to real blackbox optimization problems.

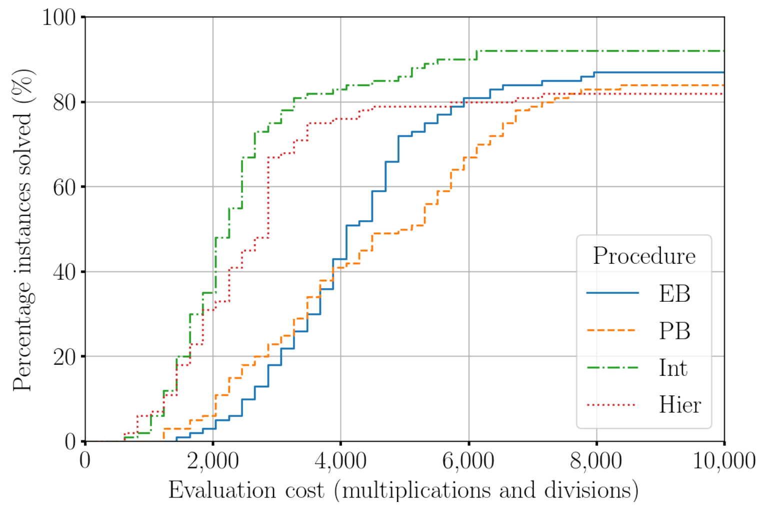

Figure 2: Data profiles showing performance of the different methods on the TCSD problem, with constraints ordered by empirical infeasibility (criterion: optimality gap of 5%).

References

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

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

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

[4] C. Audet, S. Le Digabel, and C. Tribes. The Mesh Adaptive Direct Search Algorithm for Granular and Discrete Variables. SIAM Journal on Optimization, 29(2):1164–1189, 2019.

[5] A.D. Belegundu and J.S. Arora. A study of mathematical programming methods for structural optimization. Part I: Theory. International Journal for Numerical Methods in Engineering, 21(9):1583–1599, 1985.

[6] P. Côté and J. Paquin. Optimization and Simulation Tools for Rio Tinto's Kemano Hydropower System. In Seventh International Megaprojects Workshop, Montréal, June 2019. Slides available at https://sites.grenadine.uqam.ca/sites/megaprojectworskhop/fr/megaprojectworkshop/documents/get_document/39.

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

[8] H. Garg. Solving structural engineering design optimization problems using an artificial bee colony algorithm. Journal of Industrial and Management Optimization, 10(3):777–794, 2014.

[9] D. Komljenovic, D. Messaoudi, A. Côté, M. Gaha, L. Vouligny, S. Alarie, A. Dems, and O. Blancke. Asset Management in Electrical Utilities in the Context of Business and Operational Complexity. In 14th WCEAM Proceedings, 2021.

[10] S. Le Digabel. Algorithm 909: NOMAD: Nonlinear Optimization with the MADS algorithm. ACM Transactions on Mathematical Software, 37(4):44:1–44:15, 2011.

[11] 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.

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