Algorithm Portfolios for Noisy Optimization

Marie-Liesse Cauwet, Jialin Liu, Rozière Baptiste, Olivier Teytaud

Noname manuscript No.
(will be inserted by the editor)

Algorithm Portfolios for Noisy Optimization

Marie-Liesse Cauwet · Jialin Liu · Baptiste Rozière · Olivier Teytaud

the date of receipt and acceptance should be inserted later

Abstract Noisy optimization is the optimization of objective functions corrupted by noise. A portfolio of solvers is a set of solvers equipped with an algorithm selection tool for distributing the computational power among them. Portfolios are widely and successfully used in combinatorial optimization.

In this work, we study portfolios of noisy optimization solvers. We obtain mathematically proved performance (in the sense that the portfolio performs nearly as well as the best of its solvers) by an ad hoc portfolio algorithm dedicated to noisy optimization. A somehow surprising result is that it is better to compare solvers with some *lag*, i.e., propose the current recommendation of best solver based on their performance *earlier* in the run. An additional finding is a principled method for distributing the computational power among solvers in the portfolio.

Keywords Black-Box Noisy Optimization · Algorithm Selection · Simple Regret

1 Introduction

Given an objective function, also termed fitness function, from a domain $\mathcal{D} \in \mathbb{R}^d$ to $\mathbb{R}$, numerical optimization or simply optimization is the research of points, also termed individuals or search points, with approximately optimum (e.g. minimum) objective function values.

Noisy optimization is the optimization of objective functions corrupted by noise. Black-box noisy optimization is the noisy counterpart of black-box optimization, i.e., functions for which no knowledge about the internal processes involved in the objective function can be exploited.

Algorithm Selection (AS) consists in choosing, in a portfolio of solvers, the one which is approximately the most efficient on the problem at hand. AS can mitigate

TAO, INRIA-CNRS-LRI, Univ. Paris-Sud,
91190 Gif-sur-Yvette, France
E-mail: firstname.lastname@lri.fr
https://tao.lri.fr
Tel: +33169157643
Fax: +33169156579

the difficulties for choosing a priori the best solver among a portfolio of solvers. This means that AS leads to an adaptive version of the algorithms. In some cases, AS outperforms all individual solvers by combining the good properties of each of them, with information sharing or with chaining, as discussed later. It can also be used for the sake of parallelization or parameter tuning, or for mitigating the impact of bad luck in randomized solvers. In this paper, we apply AS to the black-box noisy optimization problem. This paper extends [12] with respect to (i) showing that the *lag* is necessary; (ii) extending experimental results; (iii) improving the convergence rates thanks to an unfair distribution of the computational budget.

1.1 Noisy optimization

Noisy optimization is a key component of machine learning, from supervised learning to unsupervised or reinforcement learning; it is also relevant in streaming applications. The black-box setting is crucial in reinforcement learning where gradients are difficult and expensive to get; direct policy search [40] usually boils down to (i) choosing a representation and (ii) applying black-box noisy optimization. Some works focus on noise models in which the noise has variance close to zero around the optimum [23]; others consider actuator noise, i.e., when the fitness values are those at search points corrupted by noise [9], as shown by Eq. 2. There are also cases in which the landscape is assumed to be rugged, and the optimization algorithm should act as a low pass filter, in order to get rid of local variations [30]. The most usual model of noise is however the case of noise in which the variance does not vanish around the optimum and the goal is to find a good search point in terms of expected fitness value [18, 13] and this is the case we will consider here.

Zero-order methods, including evolution strategies [8] and derivative-free optimization [14] are natural solutions in the black-box setting; as they do not use gradients, they are not affected in such a setting. However, the noise has an impact even on such methods [3, 34], possibly mitigated by increasing the population size or averaging multiple evaluations of each search point. Using surrogate models [24] reduces the impact of noise by sharing information over the domain. Surrogate models are also a step towards higher order methods; even in black-box scenarios: a Hessian can be approximated thanks to observed fitness values and statistical learning of a surrogate model.

It is known [18, 35] that stochastic gradient by finite differences (finite differences at each iteration or by averaging over multiple iterations) can provide tight convergence rates (see tightness in [13]) in the case of an additive noise with constant variance. Fabian, in [17], has also tested the use of second order information (Hessian) approximated by finite differences.

In this paper, our portfolio will be made of the following algorithms: (i) an evolution strategy; (ii) a first-order method using gradients estimated by finite differences; (iii) a second-order method using a Hessian matrix also approximated by finite differences. We present these methods in more details in Appendix A.

1.2 Algorithm selection

Combinatorial optimization is probably the most classical application domain for AS [27]. However, machine learning is also a classical test case [41]; in this case, AS is sometimes referred to as meta-learning [1].

No free lunch. [44] claims that it is not possible to do better, on average (uniform average) on all optimization problems from a given finite domain to a given finite codomain. This implies that no AS can outperform existing algorithms on average on this uniform probability distribution of problems. Nonetheless, reality is very different from a uniform average of optimization problems, and AS does improve performance in many cases.

Chaining and information sharing. Algorithm chaining [10] means switching from one solver to another during the AS run. More generally, a *hybrid* algorithm is a combination of existing algorithms by any means [42]. This is an extreme case of sharing. Sharing consists in sending information from some solvers to other solvers; they communicate in order to improve the overall performance.

Static portfolios & parameter tuning. A portfolio of solvers is usually static, i.e., combines a finite number of given solvers. SATzilla is probably the most well known portfolio method, combining several SAT-solvers [46]. Samulowitz and Memisevic have pointed out in [33] the importance of having "orthogonal" solvers in the portfolio, so that the set of solvers is not too large, but approximates as far as possible the set of all feasible solvers. AS and parameter tuning are combined in [45]; parameter tuning can be viewed as an AS over a large but structured space of solvers. We refer to [27] and references therein for more information on parameter tuning and its relation to AS; this is beyond the scope of this paper.

Fair or unfair distribution of computation budgets. In [31], different strategies are compared for distributing the computation time over different solvers. The first approach consists in running all solvers during a finite time, then selecting the best performing one, and then keeping it for all the remaining time. Another approach consists in running all solvers with the same time budget independently of their performance on the problem at hand. Surprisingly enough, they conclude that uniformly distributing the budget is a good and robust strategy. The situation changes when a training set is available, and when we assume that the training set is relevant for the future problems to be optimized; [25], using a training set of problems for comparing solvers, proposes to use 90% of the time allocated to the best performing solver, the other 10% being equally distributed among other solvers. In [19, 20], it is proposed to use 50% of the time budget for the best solver, 25% for the second best, and so on. Some AS algorithms [20, 2] do not need a separate training phase, and perform entirely online solver selection; a weakness of this approach is that it is only possible when a large enough budget is available, so that the training phase has a minor cost. A portfolio algorithm, namely Noisy Optimization Portfolio Algorithm (NOPA), designed for noisy optimization solvers, and which distributes uniformly the computational power among them, is proposed in [12]. We extend it to INOPA (Improved NOPA), which is allowed

to distribute the budget in an unfair manner. It is proved that INOPA reaches the same convergence rate as the best solver, within a small (converging to 1) multiplicative factor on the number of evaluations, when there is a unique optimal solver - thanks to a principled distribution of the budget into (i) running all the solvers; (ii) comparing their results; (iii) running the best performing one. The approach is anytime, in the sense that the computational budget does not have to be known in advance.

Parallelism. We refer to [22] for more on parallel portfolio algorithms (though not in the noisy optimization case). Portfolios can naturally benefit from parallelism; however, the situation is different in the noisy case, which is highly parallel by nature (as noise is reduced by averaging multiple resamplings[1]).

Best solver first. [31] point out the need for a good ordering of solvers, even if it has been decided to distribute nearly uniformly the time budget among them: this improves the anytime behavior. For example, they propose, within a given scheduling with same time budget for each optimizer, to use first the best performing solver. We will adapt this idea to our context; this leads to INOPA, improved version of NOPA.

Bandit literature. During the last decade, a wide literature on bandits [28,6,11] has proposed many tools for distributing the computational power over stochastic options to be tested. The application to our context is however far from being straightforward. In spite of some adaptations to other contexts (time varying as in [26] or adversarial [21,7]), and maybe due to strong differences such as the very non-stationary nature of bandit problems involved in optimization portfolios, these methods did not, for the moment, really find their way to AS. Another approach consists in writing this bandit algorithm as a meta-optimization problem; [38] applies the differential evolution algorithm [39] to some non-stationary bandit problem, which outperforms the classical bandit algorithm on an AS task.

The main contributions of this paper can be summarized as follows. First, we prove positive results for a portfolio algorithm, termed NOPA, for AS in noisy optimization. Second, we design a new AS, namely INOPA, which (i) gives the priority to the best solvers when distributing the computational power; (ii) approximately reaches the same performance as the best solver; (iii) possibly shares information between the different solvers. We then prove the requirement of selecting the solver that was apparently the best *some time before* the current iteration - a phenomenon that we term the *lag*. Finally, we provide some experimental results.

1.3 Outline of this paper

Section 2 describes the algorithms under consideration and provides theoretical results. Section 3 is dedicated to experimental results. Section 4 concludes.

[1] "Resamplings" means that the stochastic objective function, also known as fitness function, is evaluated several times at the same search point. This mitigates the effects of noise.

2 Algorithms and analysis

Section 2.1 introduces some notations. Section 2.2 provides some background and criteria. Section 2.3 describes two portfolio algorithms, one with fair distribution of budget among solvers and one with unfair distribution of budget. Section 2.4 then provides theoretical guarantees.

2.1 Notations

In this paper, $\mathbb{N}^* = \{1, 2, 3, \dots\}$, "a.s." stands for "almost surely", i.e., with probability 1, and "s.t." stands for "such that". Appendix B is a summary of notations. If X is a random variable, then $(X^{(1)}, X^{(2)}, \dots)$ denotes a sample of independent identically distributed random variables, copies of X. $o(.), O(.), \Theta(.)$ are the standard Landau notations. $\mathcal{N}$ denotes a standard Gaussian distribution, in dimension given by the context.

Let $f : \mathcal{D} \to \mathbb{R}$ be a noisy function. f is a random process, and equivalently it can be viewed as a mapping $(x, \omega) \mapsto f(x, \omega)$ where $x \in \mathcal{D}$ and ω is a random variable independently sampled at each call to f. The user can only choose x. For short, we will use the notation $f(x)$. The reader should keep in mind that this function is stochastic. $\hat{\mathbb{E}}_s [f(x)]$ denotes the empirical evaluation of $\mathbb{E}[f(x)]$ over $s \in \mathbb{N}^*$ resamplings, i.e., $\hat{\mathbb{E}}_s [f(x)] = \frac{1}{s} \sum_{j=1}^{s} (f(x))^{(j)}$.

2.2 Definitions and Criteria

A black-box noisy optimization solver, here referred to as a solver, is a program which aims at finding the minimum x^* of $x \mapsto \mathbb{E}f(x)$, thanks to multiple black-box calls to the unknown function f. The expectation operator $\mathbb{E}$ shows that we assume that the noise is additive and unbiased (Eq. 1); the "real", noise-free, fitness function is the expectation of the noisy fitness function. This is not necessarily the case for e.g. actuator noise, as in Eq. 2:

$$f(x, \omega) = f(x) + \omega; \tag{1}$$

$$f(x, \omega) = f(x + \omega). \tag{2}$$

The portfolio algorithm has the same goal, and can use $M \in \{2, 3, \dots\}$ different given solvers. A good AS tool should ensure that it is nearly as efficient as the best of the individual solvers[2], for any problem in some class of interest.

Simple regret criterion. In the black-box setting, let us define :

- x_n the n^{th} search point at which the objective function (also termed fitness function) is evaluated;
- $\tilde{x}_n$ the point that the solver recommends as an approximation of the optimum after having evaluated the objective function at $x_1, \dots, x_n$ (i.e., after spending n evaluations from the budget).

[2] A solver is termed "individual solver" when it is not a portfolio. In this paper, unless stated otherwise, a "solver" is an "individual solver".

Some algorithms make no difference between x_n and $\tilde{x}_n$, but in the general case of a noisy optimization setting the difference matters [16,18,35].

The Simple Regret (SR) for noisy optimization is expressed in terms of objective function values, as follows:

$$SR_n = \mathbb{E}\left(f(\tilde{x}_n) - f(x^*)\right). \tag{3}$$

SR_n is the simple regret after n evaluations; n is then the budget. The $\mathbb{E}$ operator refers to the ω part, i.e., with complete notations,

$$SR_n = \mathbb{E}_\omega\left(f(\tilde{x}_n, \omega) - f(x^*, \omega)\right).$$

In many cases, it is known that the simple regret has a linear convergence in a log-log scale [18,13,16]. Therefore we will consider this slope. The slope of the simple regret is then defined as

$$s(SR) = \lim_{n\to\infty} \frac{\log(SR_n)}{\log(n)}, \tag{4}$$

where the limit holds almost surely, since SR_n is a random variable.

For example, the gradient method proposed in [18] (approximating the gradient by finite differences) reaches a simple regret slope arbitrarily close to -1 on sufficiently smooth problems, for an additive centered noise, without assuming variance decreasing to zero around the optimum.

Simple regret criterion for portfolio. For a portfolio algorithm in the black-box setting, $\forall i \in \{1, \ldots, M\}$, $\tilde{x}_{i,n}$ denotes the point

- that the solver number i *recommends* as an approximation of the optimum;
- after this solver has spent n evaluations from the budget.

Similarly, the simple regret given by Equation 3 corresponding to solver number i after n evaluations (i.e., after solver number i has spent n evaluations), is denoted by $SR_{i,n}$. For $n \in \mathbb{N}^*$, i_n^* denotes the solver chosen by the selection algorithm when there are at most n evaluations per solver.[3]

Another important concept is the difference between the two kinds of terms in the regret of the portfolio. We distinguish these two kinds of terms in the next two definitions:

Definition 1 (Solvers' regret) The solvers' regret with index n, denoted by $SR_n^{Solvers}$, is the minimum simple regret among the solvers after at most n evaluations each, i.e., $SR_n^{Solvers} = \min_{i\in\{1,\ldots M\}} SR_{i,n}$.

Definition 2 (Selection regret) The selection regret with index n, denoted by $SR_n^{Selection}$ includes the additional regret due to mistakes in choosing among these M solvers after at most n evaluations each, i.e., $SR_n^{Selection} = \mathbb{E}\left(f(\tilde{x}_{i_n^*,n}) - f(x^*)\right)$.

Similarly, $\Delta_{i,n}$ quantifies the regret for choosing solver i at iteration n.

[3] This is not uniquely defined, as there might be several time steps at which the maximum number of evaluations in a solver is n; however, the results in the rest of this paper are independent of this subtlety.

Definition 3 For any solver $i \in \{1, \ldots, M\}$ and any number of evaluations $n \in \mathbb{N}^*$, we denote by $\Delta_{i,n}$ the quantity: $\Delta_{i,n} = SR_{i,n} - \min_{j \in \{1,\ldots,M\}} SR_{j,n}$.

Finally, we consider a function that will be helpful for defining our portfolio algorithms.

Definition 4 (lag function) A lag function $\text{LAG} : \mathbb{N}^* \to \mathbb{N}^*$ is a non-decreasing function such that for all $n \in \mathbb{N}^*$, $\text{LAG}(n) \leq n$.

2.3 Portfolio algorithms

In this section, we present two AS methods. A first version, in Section 2.3.1, shares the computational budget uniformly; a second version has an unfair sharing of computation budget, in Section 2.3.2.

2.3.1 Simple Case : Uniform Portfolio NOPA

We present in Algorithm 1 a simple noisy optimization portfolio algorithm (NOPA) which does not apply any sharing and distributes the computational budget equally over the noisy optimization solvers.

Algorithm 1 Noisy Optimization Portfolio Algorithm (NOPA).

Require: noisy optimization solvers $Solver_1, Solver_2, \ldots, Solver_M$
Require: a *lag* function $\text{LAG} : \mathbb{N}^* \mapsto \mathbb{N}^*$ ▷ As in Definition 4
Require: a non-decreasing integer sequence $r_1, r_2, \ldots$ ▷ Periodic comparisons
Require: a non-decreasing integer sequence $s_1, s_2, \ldots$ ▷ Number of resamplings
$n \leftarrow 1$ ▷ Number of selections
$m \leftarrow 1$ ▷ NOPA's iteration number
$i^* \leftarrow null$ ▷ Index of recommended solver
$x^* \leftarrow null$ ▷ Recommendation
while budget is not exhausted **do**
  if $m \geq r_n$ **then**
    $i^* = \underset{i \in \{1,\ldots,M\}}{\arg\min}\ \hat{\mathbb{E}}_{s_n}[f(\tilde{x}_{i,\text{LAG}(r_n)})]$ ▷ Algorithm selection
    $n \leftarrow n + 1$
  else
    for $i \in \{1, \ldots, M\}$ **do**
      Apply one evaluation for $Solver_i$
    end for
    $m \leftarrow m + 1$
  end if
  $x^* = \tilde{x}_{i^*,m}$ ▷ Update recommendation
end while

In this NOPA algorithm, we compare, at iteration r_n, recommendations chosen at iteration $\text{LAG}(r_n)$, and this comparison is based on s_n resamplings, where n is the number of algorithm selection steps. We have designed the algorithm as follows:

- **A stable choice of solver:** The selection algorithm follows the recommendation of the same solver i^* at all iterations in $\{r_n, \ldots, r_{n+1} - 1\}$. This choice is based on comparisons between old recommendations (through the *lag* function LAG).

– **The chosen solver updates are taken into account.** For iteration indices $m < p$ in $\{r_n, \dots, r_{n+1} - 1\}$, the portfolio chooses the same solver i^*, but does not necessarily recommends the same point because possibly the solver changes its recommendation, i.e., possibly $\tilde{x}_{i^*,m} \neq \tilde{x}_{i^*,p}$.

Effect of the lag. Due to the LAG(.) function, we compare the $\tilde{x}_{i,\text{LAG}(r_n)}$ (for $i \in \{1, \dots, M\}$), and not the $\tilde{x}_{i,r_n}$. This is the key point of this algorithm. Comparing the $\tilde{x}_{i,\text{LAG}(r_n)}$ is much cheaper than comparing the $\tilde{x}_{i,r_n}$, because the fitness values are not yet that good at iteration $\text{LAG}(r_n)$, so they can be compared faster - i.e., with less evaluations - than recommendations at iteration r_n. We will make this more formal in Section 2.4, and see under which assumptions this lag has more pros than cons, namely when algorithms have somehow sustained rates. In addition, with lag, we can define INOPA, which saves up significant parts of the computation time.

The first step for formalizing this is to understand the two different kinds of evaluations in portfolio algorithms for noisy optimization. Contrarily to noise-free settings, comparing recommendations requires a dedicated budget, which is far from negligible. It follows that there are two kinds of evaluations:

– **Portfolio budget (Algorithm 1, Lines 10-12):** this corresponds to the M evaluations per iteration, dedicated to running the M solvers (one evaluation per solver and per iteration).
– **Comparison budget (Algorithm 1, Line 7):** this corresponds to the s_n evaluations per solver at the n^{th} algorithm selection. This is a key difference with deterministic optimization. In deterministic optimization, this budget is zero as the exact fitness value is readily available.

We have $M \cdot r_n$ evaluations in the portfolio budget for the first r_n iterations. We will see below (Section 2.4) conditions under which the other costs (i.e. comparison costs) can be made negligible, whilst preserving the same regret as the best of the M solvers.

2.3.2 INOPA: Improved NOPA, with unequal budget

Algorithm 2 proposes a variant of NOPA, which distributes the budget in an unfair manner. The solvers with good performance receive a greater budget. The algorithm is designed so that it mimics the behavior of NOPA, but without spending the evaluations which are useless for the moment, given the lag - i.e. we use the fact that evaluations prior to the lagged index are useless except for the selected algorithm.

2.4 Theoretical analysis

We here show

– a bound on the performance of NOPA (Section 2.4.2);
– a bound on the performance of INOPA (Section 2.4.3);
– that the *lag* term is necessary (Section 2.4.4).

Algorithm 2 Improved Noisy Optimization Portfolio Algorithm (INOPA).

Require: noisy optimization solvers $Solver_1, Solver_2, \ldots, Solver_M$
Require: a *lag* function $\text{LAG} : \mathbb{N}^* \mapsto \mathbb{N}^*$ ▷ Refer to Definition 4
Require: a non-decreasing positive integer sequence $r_1, r_2, \ldots$ ▷ Periodic comparisons
Require: a non-decreasing integer sequence $s_1, s_2, \ldots$ ▷ Number of resamplings
$n \leftarrow 1$ ▷ Number of selections
$m \leftarrow 1$ ▷ NOPA's iteration number
$i^* \leftarrow null$ ▷ Index of recommended solver
$x^* \leftarrow null$ ▷ Recommendation
while budget is not exhausted **do**
  if $m \geq \text{LAG}(r_n)$ or $i^* = null$ **then**
    $i^* = \underset{i \in \{1,\ldots,M\}}{\arg\min}\ \hat{\mathbb{E}}_{s_n}[f(\tilde{x}_{i,\text{LAG}(r_n)})]$ ▷ Algorithm selection
    $m' \leftarrow r_n$
    while $m' < r_{n+1}$ **do**
      Apply one evaluation to solver i^*
      $m' \leftarrow m' + 1$
      $x^* = \tilde{x}_{i^*,m'}$ ▷ Update recommendation
    end while
    $n \leftarrow n + 1$
  else
    for $i \in \{1, \ldots, M\} \backslash i^*$ **do**
      Apply $\text{LAG}(r_n) - \text{LAG}(r_{n-1})$ evaluations for $Solver_i$
    end for
    $m \leftarrow m + 1$
  end if
end while

2.4.1 Preliminary

We define 2 extra properties which are central in the proof.

Definition 5 ($\mathbf{P_{as}^{(i)}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$) For any solver $i \in \{1, \ldots, M\}$, for some positive sequence $(\epsilon_n)_{n\in\mathbb{N}^*}$, we define $\mathbf{P_{as}^{(i)}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$:

$$P_{as}^{(i)}((\epsilon_n)_{n\in\mathbb{N}^*}) : a.s. \quad \exists n_0,\ \forall n_1 \geq n_0,\ \Delta_{i,n_1} < 2\epsilon_{n_1} \implies \forall n_2 \geq n_1,\ \Delta_{i,n_2} < 2\epsilon_{n_2}.$$

Informally speaking, if $\mathbf{P_{as}^{(i)}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true, then almost surely for a large enough number of evaluations, the difference between the simple regret of solver $i \in \{1, \ldots, M\}$ and the optimal simple regret is either always at most $2\epsilon_n$ or always larger - there is no solver infinitely often alternatively at the top level and very weak.

Definition 6 ($\mathbf{P_{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$) For some positive sequence $(\epsilon_n)_{n\in\mathbb{N}^*}$, we define $\mathbf{P_{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ as follows:

$$\forall i \in \{1, \ldots, M\},\ \mathbf{P_{as}^{(i)}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*}) \text{ holds.}$$

Remark 1 In Definitions 5 and 6, we might choose slightly less restrictive definitions, for which the inequalities only hold for integers n such that $\exists i$, $\text{LAG}(r_i) = n$ or $r_i = n$.

Definitions above can be applied in a very general setting. The simple regret of some noisy optimization solvers, for instance Fabian's algorithm, is almost surely

$SR_n \leq (1+o(1))\frac{C}{n^\alpha}$ after $n \in \mathbb{N}^*$ evaluations (C is a constant), for some constant $\alpha > 0$ arbitrarily close to 1. This result is proved in [18], with optimality proved in [13]. For RSAES, introduced below, the proof is given in [4]. For noisy variants of Newton's algorithm, one can refer to [36,37,5].

We prove the following proposition for such a case; it will be convenient for illustrating "abstract" general results to standard noisy optimization frameworks.

Proposition 1 *Assume that each solver $i \in \{1,\dots,M\}$ has almost surely simple regret $(1+o(1))\frac{C_i}{n^{\alpha_i}}$ after $n \in \mathbb{N}^*$ evaluations.*

We define C, α^*, C^*:

$$C = \frac{1}{3}\min\left\{|C_i - C_j| \mid 1 \leq i,j \leq M; C_i - C_j \neq 0\right\}. \tag{5}$$

$$\alpha^* = \max_{i\in\{1,\dots,M\}} \alpha_i. \tag{6}$$

$$C^* = \min_{i\in\{1,\dots,M\} \ s.t. \ \alpha_i=\alpha^*} C_i. \tag{7}$$

We also define the set of optimal solvers:

$$\begin{aligned} SetOptim &= \{i \in \{1,\dots,M\} | \alpha_i = \alpha^*\} \\ and\ SubSetOptim &= \{i^* \in SetOptim | C_{i^*} = C^*\} &(8)\\ &= \{i \in \{1,\dots,M\} | \alpha_i = \alpha^* \ and \ C_i = C^*\}. &(9)\end{aligned}$$

With these notations, if almost surely, $\forall i \in \{1,\dots,M\}$, the simple regret for solver i after $n \in \mathbb{N}^*$ evaluations is $SR_{i,n} = (1+o(1))\frac{C_i}{n^{\alpha_i}}$, then $\mathbf{P_{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true with ϵ_n defined as follows:

$$\epsilon_n = \frac{C}{n^{\alpha^*}}. \tag{10}$$

Moreover, if $i_0 \in \{1,\dots,M\}$ satisfies: $(\exists n_0 \in \mathbb{N}^*, \ \forall n \geq n_0, \ \Delta_{i_0,n} \leq 2\epsilon_n)$, then $i_0 \in SubSetOptim$.

Informally speaking, this means that if the solver i_0 is close, in terms of simple regret, to an optimal solver (i.e., a solver matching α^* and C^* in Equations 6 and 7), then it also has an optimal slope ($\alpha_{i_0} = \alpha^*$) and an optimal constant ($C_{i_0} = C^*$).

Proof For any solver $i \in \{1,\dots,M\}$ and any solver $i^* \in SubSetOptim$,

$$SR_{i,n} - SR_{i^*,n} = (1+o(1))\frac{C_i}{n^{\alpha_i}} - (1+o(1))\frac{C^*}{n^{\alpha^*}}. \tag{11}$$

By Equations 10 and 11,

$$\frac{SR_{i,n} - SR_{i^*,n}}{\epsilon_n} = \frac{C_i}{C}\cdot n^{\alpha^*-\alpha_i}\cdot(1+o(1)) - \frac{C^*}{C}\cdot(1+o(1)). \tag{12}$$

– If $i \notin SetOptim$, i.e., $\alpha_i < \alpha^*$, the first term in Equation 12 tends to ∞, which leads to

$$\lim_{\alpha_i<\alpha^*, n\to\infty} \frac{SR_{i,n} - SR_{i^*,n}}{\epsilon_n} = \infty.$$

So for all $i \notin SetOptim$, $\exists n_0 \in \mathbb{N}^*$ s.t. $\forall n \geq n_0$, $\Delta_{i,n} = SR_{i,n} - \min_{j\in\{1,\dots,M\}} SR_{j,n} > 2\epsilon_n$ and, therefore, $\mathbf{P}^{(\mathbf{i})}_{\mathbf{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true.

– If $i \in SetOptim$, i.e., $\alpha_i = \alpha^*$, Equation 12 becomes

$$\frac{SR_{i,n} - SR_{i^*,n}}{\epsilon_n} = \frac{C_i - C^*}{C} + \frac{C_i}{C}o(1) - \frac{C^*}{C}o(1)$$

and therefore

$$\lim_{n\to\infty} \frac{SR_{i,n} - SR_{i^*,n}}{\epsilon_n} = \frac{C_i - C^*}{C}.$$

– If $i \in SubSetOptim$, i.e., $C_i = C^*$, $\lim_{n\to\infty} \frac{SR_{i,n}-SR_{i^*,n}}{\epsilon_n} = 0$. Therefore, $\mathbf{P}^{(\mathbf{i})}_{\mathbf{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true.
– if $i \notin SubSetOptim$, $\lim_{n\to\infty} \frac{SR_{i,n}-SR_{i^*,n}}{\epsilon_n} \geq 3$ by definition of C (Equation 5). Therefore, $\mathbf{P}^{(\mathbf{i})}_{\mathbf{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true.

So for all $i \in \{1, \ldots, M\}$, $\mathbf{P}^{(\mathbf{i})}_{\mathbf{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ is true, hence $\mathbf{P}_{\mathbf{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ holds. Moreover, it shows that $\exists n_0 \in \mathbb{N}^*$, $\forall n \geq n_0$, $SR_n^{Solvers} = \min_{j\in\{1,\ldots,M\}} SR_{j,n} = SR_{j^*,n}$ where $j^* \in SubSetOptim$.

□

2.4.2 The $\log(M)$*-shift for NOPA*

We can now enunciate the first main theorem, stating that there is, with fair sharing of the budget as in NOPA, a $\log(M)$-shift, i.e., on a log-log scale (x-axis equal to the number of evaluations and y-axis equal to the log of the simple regret), the regret of the portfolio is just shifted by $\log(M)$ on the x-axis.

Theorem 1 (Regret of NOPA: the $\log(M)$ **shift)** *Let* $(r_n)_{n\in\mathbb{N}^*}$ *and* $(s_n)_{n\in\mathbb{N}^*}$ *be two non-decreasing integer sequences. Assume that:*

– $\forall x \in \mathcal{D},\ Var\ f(x) \leq 1$*;*
– *for some positive sequence* $(\epsilon_n)_{n\in\mathbb{N}^*}$*,* $\mathbf{P}_{\mathbf{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ *(Definition 6) is true. Then, there exists* n_0 *such that:*

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\epsilon_{r_n} \tag{13}$$

$$\textit{with probability at least } 1 - \frac{M}{s_n \epsilon^2_{\mathrm{LAG}(r_n)}}$$

$$\textit{after } e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right) \textit{ evaluations.}$$

Moreover, if (s_n), $\mathrm{LAG}(n)$, (r_n) *and* (ϵ_n) *satisfy* $\sum_{j=1}^{\infty} \frac{1}{s_j \epsilon^2_{\mathrm{LAG}(r_j)}} < \infty$*, then, almost surely, there exists* n_0 *such that:*

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\epsilon_{r_n} \tag{14}$$

$$\textit{after } \ e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right) \textit{ evaluations.}$$

Remark 2 Please notice that Equation 13 holds *with a given probability* whereas Equation 14 holds *almost surely*. The almost sure convergence in the assumption is proved for some noisy optimization algorithms [18].

Proof First, the total number of evaluations, up to the construction of $\tilde{x}_{i^*_{r_n},r_n}$ at iteration r_n, is $e_n = M\left(r_n + \sum_{i=1}^n s_i\right)$; at this point, each solver has spent r_n evaluations.

Step 1: Proof of Equation 13.

By Chebyshev's inequality, for a given $i \in \{1,\dots,M\}$,

$$\mathbb{P}(|\mathbb{E}\left[f(\tilde{x}_{i,\text{LAG}(r_n)})\right] - \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i,\text{LAG}(r_n)})\right]| > \epsilon_{\text{LAG}(r_n)}) < \frac{Var\ f\left(\tilde{x}_{i,\text{LAG}(r_n)}\right)}{s_n \epsilon^2_{\text{LAG}(r_n)}} \leq \frac{1}{s_n \epsilon^2_{\text{LAG}(r_n)}}.$$

By union bound,

$$\mathbb{P}(\exists i \in \{1,\dots,M\}; |\mathbb{E}\left[f(\tilde{x}_{i,\text{LAG}(r_n)})\right] - \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i,\text{LAG}(r_n)})\right]| > \epsilon_{\text{LAG}(r_n)}) < \frac{M}{s_n \epsilon^2_{\text{LAG}(r_n)}}.$$

With notation $i^* = i^*_{r_n} = \underset{i\in\{1,\dots,M\}}{\arg\min}\ \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i,\text{LAG}(r_n)})\right]$, it follows that, with probability $1 - \frac{M}{s_n \epsilon^2_{\text{LAG}(r_n)}}$:

$$\begin{aligned}
\mathbb{E}\left[f(\tilde{x}_{i^*,\text{LAG}(r_n)})\right] &< \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{i^*,\text{LAG}(r_n)})\right] + \epsilon_{\text{LAG}(r_n)};\\
\mathbb{E}\left[f(\tilde{x}_{i^*,\text{LAG}(r_n)})\right] &< \hat{\mathbb{E}}_{s_n}\left[f(\tilde{x}_{j,\text{LAG}(r_n)})\right] + \epsilon_{\text{LAG}(r_n)},\ \forall j \in \{1,\dots,M\};\\
\mathbb{E}\left[f(\tilde{x}_{i^*,\text{LAG}(r_n)})\right] &< \mathbb{E}\left[f(\tilde{x}_{j,\text{LAG}(r_n)})\right] + 2\epsilon_{\text{LAG}(r_n)},\ \forall j \in \{1,\dots,M\};\\
\mathbb{E}\left[f(\tilde{x}_{i^*,\text{LAG}(r_n)})\right] - \mathbb{E}\left[f(x^*)\right] &< \min_{j\in\{1,\dots,M\}} SR_{j,\text{LAG}(r_n)} + 2\epsilon_{\text{LAG}(r_n)};
\end{aligned}$$

So, with probability at least $1 - \frac{M}{s_n \epsilon^2_{\text{LAG}(r_n)}}$,

$$\Delta_{i^*,\text{LAG}(r_n)} < 2\epsilon_{\text{LAG}(r_n)}. \tag{15}$$

Using $\mathbf{P_{as}}((\epsilon_{\mathbf{n}})_{\mathbf{n}\in\mathbb{N}^*})$, Equation 15 yields $\Delta_{i^*,r_n} < 2\epsilon_{r_n}$ for $\text{LAG}(r_n)$ large enough, which is the expected result.

Step 2: Proof of Equation 14.

We denote by E_n the event "$\Delta_{i^*,r_n} \geq 2\epsilon_{r_n}$" (equivalent to $SR^{Selection}_{r_n} \geq SR^{Solver}_{r_n} + 2\epsilon_{r_n}$). By Equation 13, there exists $n_0 \in \mathbb{N}^*$ such that, $\forall n \geq n_0$, $\mathbb{P}(E_n) \leq \frac{M}{s_n \epsilon^2_{\text{LAG}(r_n)}}$.

Therefore

$$\sum_{j=1}^{\infty} \mathbb{P}(E_j) \leq \sum_{j=1}^{n_0-1} \mathbb{P}(E_j) + M \sum_{j=n_0}^{\infty} \frac{1}{s_j \epsilon^2_{\text{LAG}(r_j)}} < \infty.$$

According to Borel-Cantelli lemma, almost surely, for n large enough,

$$SR^{Selection}_{r_n} < SR^{Solver}_{r_n} + 2\epsilon_{r_n}$$

and the number of evaluations is still $e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^n \frac{s_i}{r_n}\right)$. □

We now use Proposition 1 to apply Theorem 1 on a classical case with almost sure convergence.

Application 1 ($\log(M)$ shift) *Assume that for any solver $i \in \{1, \dots, M\}$, the simple regret after $n \in \mathbb{N}^*$ evaluations is $SR_{i,n} = (1 + o(1))\frac{C_i}{n^{\alpha_i}}$. We define $\epsilon_n = \frac{C}{n^{\alpha^*}}$ (where C and α^* are defined as in Equations 5 and 6). Assume that $\forall x \in \mathcal{D}$, $Var\ f(x) \leq 1$ and that (s_n), $(\text{LAG}(n))$ and (r_n) satisfy:*

$$\sum_{j=1}^{\infty} \frac{1}{s_j \epsilon^2_{\text{LAG}(r_j)}} < \infty$$
$$\text{and } \sum_{i=1}^{n} s_i = o(r_n).$$

Then, almost surely,

i) for n large enough, $SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\epsilon_{r_n}$ after $e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right)$ function evaluations;

ii) for n large enough, $SR_{r_n}^{Selection} \leq \max_{i \in SubSetOptim} SR_{i,r_n}$ after $e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right)$ function evaluations;

iii) the slope of the selection regret verifies $\lim_{n\to\infty} \frac{\log(SR_{r_n}^{Selection})}{\log(e_n)} = -\alpha^*$.

$SR_{r_n}^{Selection}$ corresponds to the simple regret at iteration r_n of the portfolio, which corresponds to $e_n = r_n \cdot M \cdot \left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right)$ evaluations in the portfolio - hence the comment "after e_n function evaluations".

Proof By Property 1 and Theorem 1, i) holds.

By Equation 15, and Property 1, $SR_{r_n}^{Selection} = SR_{i,r_n}$, with $i \in SubSetOptim$ and ii) follows. We obtain:

$$\text{a.s. } \log(SR_{r_n}^{Selection}) = \log(SR_{i,r_n}), \text{ where } i \in SubSetOptim.$$

By the definition of $SubSetOptim$ (Equation 9):

$$\begin{aligned}\lim_{n\to\infty} \frac{\log(SR_{r_n}^{Selection})}{\log(e_n)} &= \lim_{n\to\infty} \frac{\log(SR_{i^*,r_n})}{\log(M) + \log(r_n) + \log\left(1 + \sum_{i=1}^{n} \frac{s_i}{r_n}\right)} \\ &= \lim_{n\to\infty} \frac{\log(SR_{i^*,r_n})}{\log(r_n)} = -\alpha^*.\end{aligned}$$

Hence iii) holds. □

Example 1 *The following parametrization matches the conditions in Application 1.*

$$\begin{aligned} r_n &= \lceil n^{3+r+r'} \rceil; \\ \text{LAG}(n) &= \lceil \log(n) \rceil; \\ s_n &= \lceil n^{1+r'} \rceil, r > 0 \text{ and } r' \geq 1, n \in \mathbb{N}^*. \end{aligned}$$

2.4.3 The $\log(M')$*-shift for INOPA*

We now show that INOPA, which distributes the budget in an unfair manner, can have an improvement over NOPA. Instead of a factor M (number of solvers in the portfolio), we get a factor M', number of approximately optimal solvers. This is formalized in the following theorem:

Theorem 2 ($\log(M')$ **shift)** *Let* $(r_n)_{n\in\mathbb{N}^*}$ *and* $(s_n)_{n\in\mathbb{N}^*}$ *two non-decreasing integer sequences. Assume that:*

- $\forall x \in \mathcal{D},\ Var\ f(x) \leq 1$*;*
- *for some positive sequence* $(\epsilon_n)_{n\in\mathbb{N}^*}$, $\mathbf{P_{as}}((\epsilon_\mathbf{n})_{\mathbf{n}\in\mathbb{N}^*})$ *(Definition 6) holds.*

We define $S = \{i | \exists n_0 \in \mathbb{N}^*, \forall n \geq n_0, \Delta_{i,n} < 2\epsilon_n\}$ *and* M' *denotes the cardinality of the set* S*, i.e.,* $M' = |S|$*. Then, there exists* n_0 *such that:*

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\epsilon_{r_n} \tag{16}$$

$$\textit{with probability at least } 1 - \frac{M}{s_n \epsilon^2_{\mathrm{LAG}(r_n)}}$$

$$\textit{after } e_n = r_n \cdot M' \cdot \left(1 + \frac{M}{M'} \sum_{i=1}^{n} \frac{s_i}{r_n}\right) + (M - M')\mathrm{LAG}(r_n) \textit{ evaluations.}$$

Then, if (s_n), $(\mathrm{LAG}(n))$, (r_n) *and* (ϵ_n) *satisfy* $\sum_{j=1}^{\infty} \frac{1}{s_j \epsilon^2_{\mathrm{LAG}(r_j)}} < \infty$, $\mathrm{LAG}(n) = o(n)$ *and* $\sum_{j=1}^{n} s_j = o(r_n)$*, then, almost surely, there exists* n_0 *such that:*

$$\forall n \geq n_0,\ SR_{r_n}^{Selection} < SR_{r_n}^{Solver} + 2\epsilon_{r_n} \tag{17}$$

$$\textit{after } e_n = r_n \cdot M' \cdot (1 + o(1)) \textit{ evaluations.}$$

Proof For a given number of comparisons n, the INOPA algorithm makes the same comparisons and recommends the same value as the NOPA algorithm. Therefore all the results in Theorem 1 still hold, hence Eqs. 16 and 17 hold - but we have to prove the number e_n of evaluations.

As the algorithm chooses a solver which is not in S a finite number of times, there exists n_1 such that, for all $n \geq n_1$, the portfolio chooses a solver in S at the n^{th} comparison. We consider $n_0 \geq n_1$ such that $\mathrm{LAG}(n_0) \geq r_{n_1}$. For $n \geq n_0$ the new number of evaluations after n comparisons is:

$$\begin{aligned} e_n &\leq M' \cdot r_n + M \cdot \sum_{i=1}^{n} s_i + (M - M')\mathrm{LAG}(r_n) \\ &= M' \cdot r_n \cdot \left(1 + \frac{M}{M'} \sum_{i=1}^{n} \frac{s_i}{r_n} + \frac{M - M'}{M'} \frac{\mathrm{LAG}(r_n)}{r_n}\right) \\ &= M' \cdot r_n \cdot (1 + o(1)) . \end{aligned}$$

□

Using Proposition 1, we apply Theorem 2 above to the case of linearly convergent optimization solvers (linear in a log-log scale, with x-axis logarithmic of the number of evaluations and y-axis logarithmic of the simple regret).

Application 2 ($\log(M')$shift) *Assume that $\forall x \in \mathcal{D}$, $Var\ f(x) \leq 1$ and for any solver $i \in \{1, \dots, M\}$, the simple regret after $n \in \mathbb{N}^*$ evaluations is $SR_{i,n} = (1 + o(1))\frac{C_i}{n^{\alpha_i}}$. We define $\epsilon_n = \frac{C}{n^{\alpha^*}}$ with C and α^* defined as in Eq. 5 and 6. If $(s_n)_{n\in\mathbb{N}^*}$, $\text{LAG}(n)_{n\in\mathbb{N}^*}$, $(r_n)_{n\in\mathbb{N}^*}$ and $(\epsilon_n)_{n\in\mathbb{N}^*}$ are chosen such that $\sum_{j=1}^{\infty} \frac{1}{s_j \epsilon^2_{\text{LAG}(r_j)}} < \infty$, $\text{LAG}(n) = o(n)$ and $\sum_{j=1}^{n} s_j = o(r_n)$, then, almost surely, there exists n_0 such that:*

i) $\forall n \geq n_0$, $SR^{Selection}_{r_n} < SR^{Solver}_{r_n} + 2\epsilon_{r_n}$ after $e_n = M' \cdot r_n(1 + o(1))$ evaluations;

ii) $\forall n \geq n_0$, $SR^{Selection}_{r_n} \leq \max_{i \in SubSetOptim} SR_{i,r_n}$ after $e_n = M' \cdot r_n(1 + o(1))$ evaluations;

iii) the slope of the selection regret verifies $\lim_{n\to\infty} \frac{\log(SR^{Selection}_{r_n})}{\log(e_n)} = -\alpha^*$.

As usual, $SR^{Selection}_{r_n}$ corresponds to the simple regret at iteration r_n of the portfolio, which corresponds to $e_n = r_n \cdot M' \cdot (1 + o(1))$ evaluations in the portfolio - hence the comment "after e_n function evaluations".

Proof See proof of Application 1. □

Example 2 *(log(M') shift) The parametrization of Example 1 also matches the assumptions of Application 2.*

2.4.4 The lag *is necessary*

In this section, we show that, if there is no *lag* (i.e., $\forall n$, $\text{LAG}(n) = n$) whenever there are only two solvers, and whenever these solvers have different slopes, the portfolio algorithm might not have a satisfactory behavior, in the sense that, in the example below, it will select infinitely often the worst solver - unless s_n is so large that the comparison budget is not small compared to the portfolio budget.

Example 3 (The *lag* is necessary) Let us consider the behavior of NOPA without *lag*. We assume the following:

- no lag: $\forall n \in \mathbb{N}^*$, $\text{LAG}(r_n) = r_n$.
- the noise is a standard normal distribution $\mathcal{N}$;
- there are $M = 2$ solvers and the two solvers of the portfolio are such that, almost surely, $SR_{i,m} = (1 + o(1))\frac{C_i}{m^{\alpha_i}}$ after $m \in \mathbb{N}^*$ evaluations, $i \in \{1, 2\}$, with $\alpha_1 = 1 - e$ and $\alpha_2 = 1 - 2e$, where $e \in [0, 0.5)$ is a constant.
- The comparison budget is moderate compared to the portfolio budget, in the sense that
$$s_n = O(r_n^{\beta}) \tag{18}$$
with $\beta \leq 2 - 4e$.

Then, almost surely, the portfolio will select the wrong solver infinitely often.

Proof Let us assume the scenario above. Let us show that infinitely often, the portfolio will choose the wrong solver. Consider $Y_{1,n}$ and $Y_{2,n}$ defined by

$$Y_{i,n} = \frac{1}{s_n} \sum_{\ell=1}^{s_n} f(\tilde{x}_{i,r_n}, w^{(i,\ell)}) = \mathbb{E}_{\omega}[f(\tilde{x}_{i,r_n}, \omega)] + Z_i,\ i \in \{1, 2\},$$

where

– The $w^{(i,\ell)}$ are independent Gaussian random variables,
– $Z_i = \frac{1}{s_n}\sum_{\ell=1}^{s_n} w^{(i,\ell)}$,
– $\tilde{x}_{i,r_n}$ is the search point recommended by solver i after r_n evaluations,

i.e., $Y_{i,n}$ is the average of s_n evaluations of the noisy fitness function at $\tilde{x}_{i,r_n}$.

We denote for all $n \in \mathbb{N}^*$,

$$\delta_n = \mathbb{E}_\omega[f(\tilde{x}_{2,r_n},\omega)] - \mathbb{E}_\omega[f(\tilde{x}_{1,r_n},\omega)] = SR_{2,r_n} - SR_{1,r_n}.$$

δ_n is a random variable, because the expectation operator operates on ω; the random dependency in $(\tilde{x}_{1,r_n}, \tilde{x}_{2,r_n})$ remains.

$$v_{1,n} = Var\ Y_{1,n} \quad \text{and} \quad v_{2,n} = Var\ Y_{2,n}.$$

Definition 7 ($\mathcal{MR}_n$) Let $\mathcal{MR}_n$ (misranking at iteration n) be the event "the portfolio chooses the wrong solver at decision step $n \in \mathbb{N}^*$".

Remark 3 From the definitions of solvers 1 and 2, solver 1 is the best in terms of simple regret. As a result, if n is big enough, a.s., we get $SR_{1,r_n} < SR_{2,r_n}$, i.e., $\mathbb{E}_\omega[f(\tilde{x}_{1,r_n},\omega)] < \mathbb{E}_\omega[f(\tilde{x}_{2,r_n},\omega)]$. Then it is straightforward that, if a.s. $\mathbb{E}_\omega[f(\tilde{x}_{2,r_n},\omega)] + Z_2 < \mathbb{E}_w[f(\tilde{x}_{1,r_n},\omega)] + Z_1$, i.e., $\delta_n < Z_1 - Z_2$, the portfolio chooses solver 2 whereas solver 1 is the best: a.s. $\mathcal{MR}_{r_n}$ occurs.

Step 1: constructing independent events related to wrong solver choices.

Let us define $\delta'_n = 2(C_2/r_n^{1-2e} - C_1/r_n^{1-e})$. We have

$$\delta'_n = O\left(\frac{C_2}{r_n^{1-2e}}\right) \tag{19}$$

Almost surely, $\delta_n = (1+o(1))\frac{C_2}{r_n^{1-2e}} - (1+o(1))\frac{C_1}{r_n^{1-e}}$, for n sufficiently large, $\delta_n < \delta'_n$.

τ_n denotes the event: "$Z_1 - Z_2 > \delta'_n$". So, almost surely, for n sufficiently large, the event $\mathcal{MR}_n$ includes the event τ_n, i.e.

$$\text{almost surely, for } n \text{ sufficiently large, } \tau_n \subset \mathcal{MR}_n. \tag{20}$$

Step 2: Almost surely, τ_n occurs infinitely often.

The τ_n are independent, so we apply the converse of Borel-Cantelli lemma. First, compute the probability of τ_n;

$$\begin{aligned} P(\tau_n) &= P\left(\sqrt{v_{1,n}+v_{2,n}}\mathcal{N} > \delta'_n\right), \\ &= P\left(\mathcal{N} > \frac{\delta'_n}{\sqrt{v_{1,n}+v_{2,n}}}\right) \end{aligned}$$

By definition of $v_{1,n}$ and $v_{2,n}$, $\exists C > 0,\ s.t.\ \sqrt{v_{1,n}+v_{2,n}} = \frac{C}{\sqrt{s_n}}$, so by Equation 18, $\exists \tilde{C} > 0,\ \frac{1}{\sqrt{v_{1,n}+v_{2,n}}} = \frac{\sqrt{s_n}}{C} \leq \tilde{C} r_n^{\beta/2}$.

By 19, $\exists C' > 0\ s.t.\ \frac{\delta'_n}{\sqrt{v_{1,n}+v_{2,n}}} \leq C' \frac{C_2}{r_n^{1-2e}} \tilde{C} r_n^{\beta/2}$, with $\beta/2 = 1-2e$. Hence $P\left(\mathcal{N} > \frac{\delta'_n}{\sqrt{v_{1,n}+v_{2,n}}}\right) \geq P\left(\mathcal{N} > D\right)$, with $D > 0$.

We get $P(\tau_n) = \Omega(1)$, as the τ_n are independent, Borel-Cantelli's lemma (converse) implies that almost surely, τ_n occurs infinitely often.

Step 3: Concluding.

Step 2 has shown that almost surely, τ_n occurs infinitely often. Equation 20 implies that this is also true for $\mathcal{MR}_n$.

Therefore, infinitely often, the wrong solver is selected. □

3 Experimental results

This section is organized as follows. Section 3.1 introduces another version of algorithms, more adapted to some particular implementations of solvers. Section 3.2 describes the different solvers contained in the portfolios and some experimental results. Section 3.3 describes the similar solvers contained in the portfolios and some experimental results. In all tables, CT refers to the computation time and NL refers to "no *lag*". s as a unit refers to seconds.

3.1 Real world constraints & introducing sharing

The real world introduces constraints. Most solvers do not allow you to run one single fitness evaluation at a time, so that it becomes difficult to have exactly the same number of fitness evaluations per solver. We will here adapt Algorithm 1 for such a case; an additional change is the possible use of "Sharing" options (i.e., sharing information between the different solvers). The proposed algorithm is detailed in Algorithm 3.

This is an adapted version of NOPA for coarse grain, i.e., in case the solvers can not be restricted to doing one fitness evaluation at a time. The adaptation of INOPA is straightforward. We now present experiments with this adapted algorithm.

3.2 Experiments with different solvers in the portfolio

In this paper, unless specified otherwise, a portfolio *with lag* means that $\forall n \in \mathbb{N}^*, \text{LAG}(n) < n$. For the portfolio *without lag*, at the n^{th} algorithm selection, we compare $\tilde{x}_{i,r_n}$ instead of $\tilde{x}_{i,\text{LAG}(r_n)}$. This means that we choose the identity as a *lag* function LAG, i.e., $\forall n \in \mathbb{N}^*, \text{LAG}(n) = n$.

For our experiments below, we use four noisy optimization solvers and portfolios of these solvers with and without information sharing:

- Solver 1: A self-adaptive (μ,λ) evolution strategy with resampling as explained in Algorithm 4, with parametrization $\lambda = 10d$, $\mu = 5d$, $K = 10$, $\zeta = 2$ (in dimension d). This solver will be termed $RSAES$ (resampling self-adaptive evolution strategy).
- Solver 2: Fabian's solver, as detailed in Algorithm 5, with parametrization $\gamma = 0.1$, $a = 1$, $c = 100$. This variant will be termed $Fabian1$.
- Solver 3: Another Fabian's solver with parametrization $\gamma = 0.49$, $a = 1$, $c = 2$. This variant will be termed $Fabian2$.

Algorithm 3 Adapted version of NOPA in real world constraints.

Require: noisy optimization solvers $Solver_1, Solver_2, \ldots, Solver_M$
Require: a *lag* function $\text{LAG} : \mathbb{N}^* \mapsto \mathbb{N}^*$ ▷ Refer to Definition 4
Require: a non-decreasing integer sequence $r_1, r_2, \ldots$ ▷ Periodic comparisons
Require: a non-decreasing integer sequence $s_1, s_2, \ldots$ ▷ Number of resamplings
Require: a boolean *sharing*
$n \leftarrow 1$ ▷ Number of selections
$i^* \leftarrow null$ ▷ Index of recommended solver
$x^* \leftarrow null$ ▷ Recommendation
$R \leftarrow 0^M$ ▷ Vector of number of evaluations
while budget is not exhausted **do**
  if $\min_{i \in \{1,\ldots,M\}} R_i \geq r_n$ **then**
    $i^* = \operatorname{arg\,min}_{i \in \{1,\ldots,M\}} \hat{\mathbb{E}}_{s_n}[f(\tilde{x}_{i,\text{LAG}(r_n)})]$ ▷ Algorithm selection
    $x^* = \tilde{x}_{i^*,R_{i^*}}$ ▷ Update recommendation
    if *sharing* **then**
      All solvers receive x^* as next iterate
    end if
    $n \leftarrow n + 1$
  else
    for $i \in \{1, \ldots, M\}$ **do**
      while $R_i < r_n$ **do**
        Apply one iteration for $Solver_i$, increase R_i by the number of evals. spent
      end while
    end for
  end if
end while
$x^* = \tilde{x}_{i^*,R_{i^*}}$ ▷ Update recommendation

- Solver 4: A version of Newton's solver adapted for black-box noisy optimization (gradients and Hessians are approximated on samplings of the objective function), as detailed in Algorithm 6, with parametrization $B = 1$, $\beta = 2$, $A = 100$, $\alpha = 4$. For short this solver will be termed $Newton$.
- NOPA NL: NOPA of solvers 1-4 *without lag*. Functions are $r_n = \lceil n^{4.2} \rceil$, $\text{LAG}(n) = n$, $s_n = \lceil n^{2.2} \rceil$ at n^{th} algorithm selection.
- NOPA: NOPA of solvers 1-4. Functions are $r_n = \lceil n^{4.2} \rceil$, $\text{LAG}(n) = \lceil n^{1/4.2} \rceil$, $s_n = \lceil n^{2.2} \rceil$ at n^{th} algorithm selection.
- NOPA+S.: NOPA of solvers 1-4, with information sharing enabled. Same functions.
- INOPA: INOPA of solvers 1-4. Same functions.
- INOPA+S.: INOPA of solvers 1-4, with information sharing enabled. Same functions.

Roughly speaking, Fabian's algorithm is aimed at dealing with $z = 0$ [18]. RSAES is designed for multimodal and/or parallel optimization; it is also competitive in the unimodal setting when $z > 0$ [32,5]. Newton's solver is excellent when there is very little noise [5]. Consistently with Equation 4, we evaluate the slope of the linear convergence in log-log scale by the logarithm of the average simple regret divided by the logarithm of the number of evaluations.

Table 1 Experiments on $f(x) = \|x\|^2 + \|x\|^z\mathcal{N}$ in dimension 2 with $z = 0, 1, 2$. Numbers in this table are slopes (Eq. 4). We see that the portfolio successfully keeps the best of each world (i.e. INOPA has nearly the same slope as the best of the solvers). Results are averaged over 50 runs. "s" as a unit refers to "seconds". *Optimal* means the optimum is reached for at least one run; the number of times the optimum was reached (over the 50 runs) is given between parentheses. The standard deviation is shown after ±. Table 2 presents the same results in dimension 15. "NL" refers to "no lag" cases, i.e., $\forall n \in \mathbb{N}^*, \text{LAG}(n) = n$.

Solver/Portfolio	Obtained slope for $d = 2$, $z = 0$				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$-.391 \pm .009$	$-.391 \pm .009$	$-.396 \pm .010$	$-.381 \pm .012$	$-.394 \pm .012$
$Fabian1$	$\mathbf{-1.188 \pm .012}$	$\mathbf{-1.188 \pm .011}$	$\mathbf{-1.217 \pm .010}$	$\mathbf{-1.241 \pm .013}$	$\mathbf{-1.265 \pm .011}$
$Fabian2$	$-.172 \pm .011$	$-.161 \pm .009$	$-.178 \pm .011$	$-.212 \pm .015$	$-.226 \pm .012$
$Newton$	$-.206 \pm .009$	$-.206 \pm .009$	$-.212 \pm .010$	$-.237 \pm .011$	$-.239 \pm .011$
NOPA NL	$-.999 \pm .047$	$-.870 \pm .061$	$-.682 \pm .064$	$-.748 \pm .066$	$-.662 \pm .067$
NOPA+S. NL	$-.210 \pm .012$	$-.230 \pm .011$	$-.243 \pm .013$	$-.260 \pm .013$	$-.255 \pm .015$
NOPA	$-.897 \pm .054$	$-.946 \pm .049$	$-.835 \pm .059$	$-.777 \pm .064$	$-.932 \pm .058$
NOPA+S.	$-.264 \pm .013$	$-.298 \pm .015$	$-.268 \pm .011$	$-.304 \pm .018$	$-.303 \pm .016$
INOPA	$-.829 \pm .069$	$-.950 \pm .062$	$-.948 \pm .055$	$-.913 \pm .058$	$-.904 \pm .055$
INOPA+S.	$-.703 \pm .058$	$-.938 \pm .056$	$-.844 \pm .055$	$-.789 \pm .055$	$-.776 \pm .055$

Solver/Portfolio	Obtained slope for $d = 2$, $z = 1$				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$-.526 \pm .013$	$-.530 \pm .016$	$-.507 \pm .012$	$-.507 \pm .017$	$-.522 \pm .014$
$Fabian1$	$-1.247 \pm .015$	$-1.225 \pm .009$	$-1.252 \pm .010$	$-1.276 \pm .011$	$-1.314 \pm .013$
$Fabian2$	$-1.785 \pm .009$	$-1.755 \pm .011$	$-1.782 \pm .015$	$-1.777 \pm .011$	$-1.738 \pm .010$
$Newton$	$\mathbf{-2.649 \pm .010}$	$\mathbf{-2.605 \pm .008}$	$\mathbf{-2.600 \pm .011}$	$\mathbf{-2.547 \pm .011}$	$-2.517 \pm .010$
NOPA NL	$-1.624 \pm .011$	$-1.600 \pm .011$	$-1.593 \pm .016$	$-1.554 \pm .013$	$-1.533 \pm .014$
NOPA+S. NL	$-1.225 \pm .013$	$-1.228 \pm .013$	$-1.281 \pm .014$	$-1.298 \pm .015$	$-1.323 \pm .017$
NOPA	$-1.925 \pm .081$	$-1.954 \pm .076$	$-1.661 \pm .070$	$-1.805 \pm .066$	$-1.694 \pm .062$
NOPA+S.	$-1.491 \pm .077$	$-1.624 \pm .080$	$-1.693 \pm .072$	$-1.632 \pm .068$	$-1.537 \pm .061$
INOPA	$-2.271 \pm .062$	$-2.330 \pm .061$	$-2.478 \pm .033$	$-2.506 \pm .047$	$\mathbf{-2.599 \pm .024}$
INOPA+S.	$-2.013 \pm .070$	$-1.927 \pm .074$	$-1.987 \pm .074$	$-2.120 \pm .081$	$-1.856 \pm .078$

Solver/Portfolio	Obtained slope for $d = 2$, $z = 2$				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$-.500 \pm .013$	$-.491 \pm .011$	$-.484 \pm .011$	$-.526 \pm .018$	$-.537 \pm .015$
$Fabian1$	$-1.233 \pm .010$	$-1.246 \pm .013$	$-1.258 \pm .011$	$-1.299 \pm .014$	$-1.310 \pm .013$
$Fabian2$	$-3.173 \pm .010$	$-3.175 \pm .009$	$-3.141 \pm .008$	$-3.120 \pm .013$	$-3.073 \pm .011$
$Newton$	$-4.146 \pm .004$	$-4.349 \pm .008$	$-4.514 \pm .004$	$-4.743 \pm .012$	$-4.973 \pm .011$
NOPA NL	$-2.911 \pm .009$	$-2.871 \pm .010$	$-2.796 \pm .011$	$-2.770 \pm .012$	$-2.717 \pm .014$
NOPA+S. NL	$-2.919 \pm .011$	$-2.818 \pm .050$	$-2.785 \pm .047$	$-2.684 \pm .056$	$-2.762 \pm .016$
NOPA	$-4.343 \pm .006$	$-4.603 \pm .013$	$-4.772 \pm .013$	**Optimal** (1)	$-5.103 \pm .011$
NOPA+S.	$-4.305 \pm .041$	$-4.573 \pm .011$	$-4.431 \pm .091$	$-4.910 \pm .048$	$-5.020 \pm .059$
INOPA	**Optimal** (1)	**Optimal** (2)	$-4.698 \pm .004$	$-4.435 \pm .007$	-4.408 ± 0
INOPA+S.	**Optimal** (1)	$-3.302 \pm .116$	**Optimal** (2)	$-4.409 \pm .008$	**Optimal** (35)

3.2.1 Experiments in unimodal case

The experiments presented in this section have been performed on

$$f(x) = \|x\|^2 + \|x\|^z\mathcal{N} \tag{21}$$

with $\mathcal{N}$ a Gaussian standard noise. $z = 2$ is the so-called multiplicative noise case, $z = 0$ is the additive noise case, $z = 1$ is intermediate. The results in dimension 2 and dimension 15 are shown in Table 1 and 2.

We see on these experiments that:

- For $z = 2$ the noise-handling version of Newton's algorithm, $Newton$, performs best among the individual solvers.
- For $z = 1$ the noise-handling version of Newton's algorithm, $Newton$, performs best in dimension 2 and the second variant of Fabian's algorithm, $Fabian2$, performs best in dimension 15.

Table 2 Experiments on $f(x) = \|x\|^2 + \|x\|^z\mathcal{N}$ in dimension 15 with $z = 0, 1, 2$. Numbers in this table are slopes (Eq. 4). We see that the portfolio successfully keeps the best of each world (INOPA has nearly the same slope as the best). Results are averaged over 50 runs. "s" as a unit refers to "seconds". *Optimal* means the optimum is reached for at least one run; the number of times the optimum was reached (over the 50 runs) is given between parentheses. The standard deviation is shown after $\pm$. "NL" refers to "no lag" cases, i.e., $\forall n \in \mathbb{N}^*, \text{LAG}(n) = n$.

Solver/Portfolio	Obtained slope for $d = 15$, $z = 0$				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	.093 ± .002	.107 ± .002	.114 ± .002	.128 ± .002	.136 ± .003
$Fabian1$	**−.825 ± .003**	**−.826 ± .003**	**−.838 ± .003**	**−.834 ± .004**	**−.835 ± .003**
$Fabian2$	.096 ± .003	.108 ± .003	.108 ± .003	.114 ± .003	.125 ± .003
$Newton$	−.055 ± .002	−.062 ± .003	−.070 ± .003	−.069 ± .003	−.071 ± .003
NOPA NL	−.512 ± .046	−.393 ± .049	−.377 ± .048	−.425 ± .049	−.380 ± .046
NOPA+S. NL	.026 ± .008	−.026 ± .021	−.082 ± .025	−.237 ± .033	−.410 ± .028
NOPA	−.757 ± .003	−.750 ± .003	−.747 ± .003	−.734 ± .013	−.705 ± .018
NOPA+S.	.039 ± .007	.019 ± .013	.016 ± .019	.005 ± .024	−.079 ± .029
INOPA	−.762 ± .024	−.768 ± .024	−.822 ± .003	−.821 ± .003	−.826 ± .003
INOPA+S.	−.484 ± .033	−.508 ± .035	−.575 ± .038	−.603 ± .036	−.499 ± .037

Solver/Portfolio	Obtained slope for $d = 15$, $z = 1$				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	.094 ± .002	.102 ± .002	.118 ± .003	.128 ± .002	.137 ± .003
$Fabian1$	−.991 ± .003	−1.004 ± .003	−1.011 ± .003	−1.020 ± .003	−1.032 ± .003
$Fabian2$	**−1.399 ± .003**	**−1.376 ± .004**	−1.339 ± .003	−1.313 ± .003	−1.274 ± .004
$Newton$	−.793 ± .099	−.787 ± .095	−.959 ± .092	−.837 ± .086	−.875 ± .078
NOPA NL	−1.226 ± .003	−1.167 ± .012	−.978 ± .013	−.949 ± .008	−.943 ± .005
NOPA+S. NL	−.771 ± .058	−.869 ± .065	−.839 ± .068	−.860 ± .060	−.756 ± .052
NOPA	−.980 ± .018	−.962 ± .013	−.937 ± .005	−.941 ± .005	−.943 ± .004
NOPA+S.	−1.012 ± .020	−1.029 ± .025	−1.019 ± .021	−1.002 ± .014	−.951 ± .010
INOPA	−1.114 ± .016	−1.268 ± .026	−1.359 ± .027	−1.393 ± .018	**−1.482 ± .026**
INOPA+S.	−1.194 ± .030	−1.250 ± .038	**−1.556 ± .030**	**−1.441 ± .041**	−1.399 ± .051

Solver/Portfolio	Obtained slope for $d = 15$, $z = 2$				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	.094 ± .003	.102 ± .002	.113 ± .003	.125 ± .003	.146 ± .002
$Fabian1$	−.991 ± .003	−1.000 ± .003	−1.016 ± .003	−1.019 ± .003	−1.037 ± .004
$Fabian2$	−2.595 ± .003	−2.546 ± .003	−2.481 ± .003	−2.413 ± .003	−2.337 ± .004
$Newton$	−2.911 ± .279	−2.763 ± .291	−2.503 ± .285	−2.420 ± .265	−2.614 ± .240
NOPA NL	−2.257 ± .002	−2.184 ± .003	−2.106 ± .003	−2.000 ± .003	−1.891 ± .003
NOPA+S. NL	−1.220 ± .117	−1.690 ± .134	−2.181 ± .175	−2.131 ± .185	−2.307 ± .157
NOPA	−2.956 ± .121	−2.664 ± .107	−2.515 ± .095	−2.466 ± .090	−2.025 ± .050
NOPA+S.	**−3.996 ± .029**	**−3.796 ± .003**	**−3.567 ± .004**	−3.294 ± .003	−2.947 ± .026
INOPA	−3.005 ± .106	−3.157 ± .123	−3.319 ± .135	**−3.528 ± .144**	**−3.751 ± .136**
INOPA+S.	−3.090 ± .003	−2.942 ± .003	−2.791 ± .004	−2.673 ± .002	−2.574 ± .003

- For $z = 0$ the first variant of Fabian's algorithm, $Fabian1$, performs best (consistently with [18]).
- The portfolio algorithm successfully reaches almost the same slope as the best of its solvers and sometimes outperforms all of them.
- Portfolio with *lag* performs better than without *lag*.
- In the case of small noise, NOPA with information sharing, termed NOPA+S., performs better than without information sharing, NOPA, in dimension 15.
- Results clearly show the superiority of INOPA over NOPA.

Incidentally, the poor behavior of RSAES on such a smooth case is not a surprise. Other experiments in Section 3.2.2 show that in multimodal cases, RSAES is by far the most efficient solver among solvers above.

3.2.2 Experiments in a multimodal setting

Experiments have been performed on a Cartpole control problem with neural network controller. The controller is a feed-forward neural network with one hidden

Table 3 Slope of simple regret for control of the "Cartpole" problem using a Neural Network policy with different numbers of neurons. This test case is multimodal. These results are averaged over 50 runs. "s" as a unit refers to "seconds". The standard deviation is shown after $\pm$ and shows the statistical significance of the results. Values close to 0 correspond to cases with no convergence to the optimum, i.e., a slope zero means that the log-log curve is horizontal. The test case is the one from [43, 15]. Consistently with these references, the optimal fitness is zero.

Solver/Portfolio	Obtained slope with 2 neurons				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$-.503 \pm .008$	$-.503 \pm .008$	$-.483 \pm .007$	$\mathbf{-.469 \pm .006}$	$\mathbf{-.465 \pm .003}$
$Fabian1$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
$Fabian2$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
$Newton$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
NOPA NL	$-.442 \pm .014$	$-.469 \pm .010$	$-.465 \pm .006$	$-.452 \pm .005$	$-.433 \pm .006$
NOPA+S. NL	$-.399 \pm .020$	$-.465 \pm .009$	$-.434 \pm .015$	$-.461 \pm .007$	$-.462 \pm .005$
NOPA	$-.480 \pm .014$	$-.465 \pm .009$	$-.466 \pm .008$	$-.430 \pm .013$	$-.431 \pm .009$
NOPA+S.	$-.461 \pm .017$	$-.436 \pm .020$	$-.475 \pm .011$	$-.431 \pm .015$	$-.415 \pm .015$
INOPA	$-.501 \pm .009$	$-.468 \pm .009$	$-.458 \pm .007$	$-.445 \pm .006$	$-.424 \pm .006$
INOPA+S.	$\mathbf{-.524 \pm .011}$	$\mathbf{-.522 \pm .007}$	$\mathbf{-.490 \pm .006}$	$\mathbf{-.469 \pm .006}$	$-.459 \pm .006$

Solver/Portfolio	Obtained slope with 4 neurons				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$\mathbf{-.517 \pm .009}$	$-.503 \pm .006$	$-.481 \pm .006$	$-.458 \pm .007$	$-.452 \pm .004$
$Fabian1$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
$Fabian2$	$.002 \pm 0$	$-.007 \pm .010$	$-.016 \pm .013$	$-.006 \pm .008$	$-.005 \pm .007$
$Newton$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
NOPA NL	$-.485 \pm .010$	$-.465 \pm .011$	$-.461 \pm .007$	$-.460 \pm .004$	$-.440 \pm .004$
NOPA+S. NL	$-.474 \pm .010$	$-.487 \pm .008$	$-.490 \pm .007$	$\mathbf{-.474 \pm .006}$	$\mathbf{-.463 \pm .004}$
NOPA	$-.491 \pm .009$	$-.477 \pm .006$	$-.459 \pm .007$	$-.434 \pm .006$	$-.423 \pm .006$
NOPA+S.	$-.505 \pm .010$	$-.504 \pm .009$	$\mathbf{-.491 \pm .007}$	$-.470 \pm .006$	$-.452 \pm .006$
INOPA	$-.481 \pm .012$	$-.480 \pm .007$	$-.429 \pm .010$	$-.423 \pm .008$	$-.408 \pm .005$
INOPA+S.	$-.506 \pm .009$	$\mathbf{-.506 \pm .008}$	$-.479 \pm .007$	$-.466 \pm .006$	$-.439 \pm .005$

Solver/Portfolio	Obtained slope with 6 neurons				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$-.496 \pm .008$	$\mathbf{-.508 \pm .008}$	$-.479 \pm .007$	$-.462 \pm .004$	$-.439 \pm .005$
$Fabian1$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
$Fabian2$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
$Newton$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
NOPA NL	$-.483 \pm .008$	$-.477 \pm .009$	$-.464 \pm .005$	$-.447 \pm .005$	$-.432 \pm .005$
NOPA+S. NL	$-.489 \pm .011$	$-.496 \pm .007$	$-.488 \pm .005$	$\mathbf{-.469 \pm .005}$	$-.464 \pm .005$
NOPA	$-.492 \pm .021$	$-.498 \pm .007$	$-.477 \pm .012$	$-.464 \pm .011$	$-.456 \pm .004$
NOPA+S.	$-.430 \pm .026$	$-.446 \pm .020$	$-.452 \pm .014$	$-.442 \pm .016$	$-.417 \pm .014$
INOPA	$-.501 \pm .010$	$-.485 \pm .010$	$-.487 \pm .006$	$-.463 \pm .006$	$-.447 \pm .003$
INOPA+S.	$\mathbf{-.517 \pm .009}$	$-.501 \pm .010$	$\mathbf{-.503 \pm .006}$	$-.468 \pm .005$	$\mathbf{-.467 \pm .003}$

Solver/Portfolio	Obtained slope with 8 neurons				
	$CT = 10s$	$CT = 20s$	$CT = 40s$	$CT = 80s$	$CT = 160s$
$RSAES$	$-.493 \pm .009$	$-.475 \pm .006$	$-.449 \pm .006$	$-.436 \pm .007$	$-.420 \pm .005$
$Fabian1$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
$Fabian2$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$-.005 \pm .007$	$.002 \pm 0$
$Newton$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$	$.002 \pm 0$
NOPA NL	$-.464 \pm .010$	$-.468 \pm .007$	$-.431 \pm .008$	$-.429 \pm .006$	$-.419 \pm .006$
NOPA+S. NL	$-.463 \pm .008$	$-.480 \pm .009$	$-.485 \pm .008$	$-.466 \pm .006$	$-.453 \pm .005$
NOPA	$-.483 \pm .011$	$-.485 \pm .009$	$-.475 \pm .006$	$-.465 \pm .005$	$-.436 \pm .005$
NOPA+S.	$-.510 \pm .009$	$\mathbf{-.498 \pm .008}$	$\mathbf{-.508 \pm .006}$	$\mathbf{-.482 \pm .005}$	$-.454 \pm .006$
INOPA	$-.488 \pm .010$	$-.463 \pm .009$	$-.455 \pm .007$	$-.422 \pm .007$	$-.426 \pm .006$
INOPA+S.	$\mathbf{-.523 \pm .008}$	$-.492 \pm .009$	$-.476 \pm .007$	$-.459 \pm .007$	$\mathbf{-.460 \pm .004}$

layer of neurons. We use the same solvers as in Section 3.2.1. The results are shown in Table 3.

We see on these experiments that:

- $RSAES$ is the most efficient individual solver.
- The portfolio algorithm successfully reaches almost the same slope as the best of its solvers.

– Sometimes, the portfolio outperforms the best of its solvers.
– Results clearly show the superiority of INOPA over NOPA.

3.3 The *lag*: experiments with different variants of Fabian's algorithm

In this section, we check if the version with *lag* disabled (i.e., $\forall n \in \mathbb{N}^*, \text{LAG}(n) = n$) can compete with the version with *lag* enabled (i.e., $\forall n \in \mathbb{N}^*, \text{LAG}(n) < n$). In previous experiments this was the case, we here focus on a case in which solvers are close to each other and check if in such a case the *lag* is beneficial.

Fabian's algorithm [18] is a gradient descent algorithm using finite differences for approximating gradients. $\frac{a}{n}$ is the step in updates, i.e., the current estimate is updated by adding $-\frac{a}{n}\nabla f$ where ∇f is the approximate gradient. ∇f is approximated by averaging multiple redundant estimates, each of them by finite differences of size $\Theta(c/n^\gamma)$. Therefore, Fabian's algorithm has 3 parameters, termed a, c and γ. In the case of approximately quadratic functions with additive noise, Fabian's algorithm can obtain a good $s(SR)$ with small $\gamma > 0$. However, a and c have an important non-asymptotic effect and the tuning of these a and c parameters is challenging. A portfolio of variants of Fabian's algorithm can help to overcome the tedious parameter tuning.

For these experiments, we consider the same noisy function as in Section 3.2.1. We use 5 noisy optimization solvers which are variants of Fabian's algorithm, as detailed in Algorithm 5, and portfolio of these solvers with and without *lag*:

– Solver 1: $Fabian1$ as used in Section 3.2.
– Solver 2: Fabian's solver with parametrization $\gamma = 0.1$, $a = 5$, $c = 100$.
– Solver 3: Fabian's solver with parametrization $\gamma = 0.1$, $a = 1$, $c = 200$.
– Solver 4: Fabian's solver with parametrization $\gamma = 0.1$, $a = 1$, $c = 1$.
– Solver 5: Fabian's solver with parametrization $\gamma = 0.1$, $a = 1$, $c = 10$.
– NOPA NL: Portfolio of solvers 1-5 *without lag*. Functions are $r_n = \lceil n^{4.2} \rceil$, $\text{LAG}(n) = n$, $s_n = \lceil n^{2.2} \rceil$ at n^{th} algorithm selection.
– NOPA: NOPA of solvers 1-5. Functions are $r_n = \lceil n^{4.2} \rceil$, $\text{LAG}(n) = \lceil n^{1/4.2} \rceil$, $s_n = \lceil n^{2.2} \rceil$ at n^{th} algorithm selection.
– NOPA+S.: NOPA of solvers 1-5, with information sharing enabled. Same functions.
– INOPA: INOPA of solvers 1-5. Same functions.
– INOPA+S.: INOPA of solvers 1-5, with information sharing enabled. Same functions.

Experiments have been performed in dimension 2 and dimension 15. These 5 variants of Fabian's algorithm have asymptotically similar performance. Table 4 compares the portfolio above without *lag*, NOPA and INOPA.

We see on these experiments that:

– The *lag* is usually beneficial, though this is not always the case.
– Again, INOPA clearly outperforms NOPA.

3.4 Discussion of experimental results

In short, experiments

Table 4 Experiments on $f(x) = \|x\|^2 + \|x\|^z\mathcal{N}$ in dimension 2 and dimension 15 with $z = 0, 1, 2$. Results are mean of 1000 runs. Solvers are various parametrizations of Fabian's algorithm (see text). "s" as a unit refers to "seconds". The standard deviation is shown after $\pm$ and shows the statistical significance of the results. We use smaller time settings - this is because here the objective function has a negligible computation time.

Portfolio	obtained slope for $d = 2$				
	$CT = 0.05s$	$CT = 0.1s$	$CT = 0.2s$	$CT = 0.4s$	$CT = 0.8s$
			$z = 0$		
NOPA NL	$-1.157 \pm .009$	$-1.223 \pm .010$	$-1.146 \pm .009$	$-1.109 \pm .009$	$-1.043 \pm .009$
NOPA+S. NL	$-1.030 \pm .009$	$-1.002 \pm .011$	$-.807 \pm .010$	$-.839 \pm .009$	$-.774 \pm .007$
NOPA	$-1.255 \pm .009$	$-1.203 \pm .009$	$-1.156 \pm .008$	$-1.145 \pm .007$	$-1.152 \pm .007$
NOPA+S.	$-1.030 \pm .009$	$-1.044 \pm .008$	$-.995 \pm .009$	$-.963 \pm .008$	$-.926 \pm .007$
INOPA	$\mathbf{-1.289 \pm .010}$	$-1.247 \pm .008$	$\mathbf{-1.201 \pm .008}$	$\mathbf{-1.188 \pm 0.008}$	$\mathbf{-1.153 \pm 0.007}$
INOPA+S.	$-1.246 \pm .010$	$\mathbf{-1.266 \pm .008}$	$-1.182 \pm .010$	-1.135 ± 0.010	-1.113 ± 0.009
			$z = 1$		
NOPA NL	$-1.529 \pm .005$	$-1.471 \pm .005$	$-1.455 \pm .004$	$-1.414 \pm .004$	$-1.381 \pm .004$
NOPA+S. NL	$-1.436 \pm .006$	$-1.401 \pm .006$	$-1.287 \pm .006$	$-1.225 \pm .005$	$-1.129 \pm .005$
NOPA	$-1.543 \pm .005$	$-1.490 \pm .004$	$-1.456 \pm .004$	$-1.416 \pm .003$	$-1.377 \pm .003$
NOPA+S.	$-1.469 \pm .005$	$-1.462 \pm .005$	$-1.377 \pm .005$	$-1.339 \pm .005$	$-1.301 \pm .004$
INOPA	$\mathbf{-1.656 \pm .004}$	$\mathbf{-1.578 \pm .005}$	$\mathbf{-1.531 \pm .004}$	$\mathbf{-1.497 \pm .005}$	$\mathbf{-1.443 \pm .004}$
INOPA+S.	$-1.638 \pm .005$	$-1.568 \pm .005$	$-1.503 \pm .005$	$-1.474 \pm .005$	$-1.425 \pm .005$
			$z = 2$		
NOPA NL	$-1.528 \pm .004$	$-1.505 \pm .005$	$-1.440 \pm .004$	$-1.394 \pm .004$	$-1.375 \pm .003$
NOPA+S. NL	$-1.456 \pm .005$	$-1.393 \pm .006$	$-1.303 \pm .005$	$-1.250 \pm .005$	$-1.168 \pm .005$
NOPA	$-1.540 \pm .005$	$-1.529 \pm .004$	$-1.439 \pm .004$	$-1.422 \pm .004$	$-1.384 \pm .003$
NOPA+S.	$-1.473 \pm .006$	$-1.450 \pm .005$	$-1.371 \pm .004$	$-1.339 \pm .004$	$-1.303 \pm .004$
INOPA	$\mathbf{-1.681 \pm .005}$	$\mathbf{-1.607 \pm .004}$	$\mathbf{-1.530 \pm .005}$	$\mathbf{-1.497 \pm .004}$	$\mathbf{-1.439 \pm .005}$
INOPA+S.	$-1.578 \pm .006$	$-1.570 \pm .006$	$-1.517 \pm .005$	$-1.465 \pm .005$	-1.434 ± 0.005

Portfolio	obtained slope for $d = 15$				
	$CT = 0.05s$	$CT = 0.1s$	$CT = 0.2s$	$CT = 0.4s$	$CT = 0.8s$
			$z = 0$		
NOPA NL	$-.673 \pm .001$	$-.688 \pm .001$	$-.699 \pm .001$	$-.761 \pm .002$	$-.779 \pm .002$
NOPA+S. NL	$-.664 \pm .001$	$-.684 \pm .001$	$-.703 \pm .001$	$-.716 \pm .001$	$-.750 \pm .001$
NOPA	$-.700 \pm .006$	$-.609 \pm .006$	$-.667 \pm .004$	$-.681 \pm .005$	$-.694 \pm .004$
NOPA+S.	$-.591 \pm .006$	$-.515 \pm .006$	$-.514 \pm .005$	$-.519 \pm .004$	$-.527 \pm .004$
INOPA	$\mathbf{-.839 \pm .001}$	$\mathbf{-.839 \pm .001}$	$\mathbf{-.841 \pm .001}$	$\mathbf{-.840 \pm .001}$	$-.839 \pm .001$
INOPA+S.	$\mathbf{-.839 \pm .001}$	$\mathbf{-.839 \pm .001}$	$\mathbf{-.841 \pm .001}$	$-.839 \pm .001$	$\mathbf{-.841 \pm .001}$
			$z = 1$		
NOPA NL	$-1.004 \pm .001$	$-.991 \pm .001$	$-.980 \pm .001$	$-.978 \pm .001$	$-1.062 \pm .001$
NOPA+S. NL	$-1.000 \pm .001$	$-.985 \pm .001$	$-.980 \pm .001$	$-.990 \pm .001$	$-1.066 \pm .001$
NOPA	$-1.154 \pm .001$	$-1.140 \pm .001$	$-1.117 \pm .001$	$-1.100 \pm .001$	$-1.086 \pm .001$
NOPA+S.	$-1.160 \pm .001$	$-1.133 \pm .001$	$-1.109 \pm .001$	$-1.084 \pm .001$	$-1.065 \pm .001$
INOPA	$-1.231 \pm .001$	$\mathbf{-1.249 \pm .001}$	$\mathbf{-1.238 \pm .001}$	$\mathbf{-1.218 \pm .001}$	$\mathbf{-1.200 \pm .001}$
INOPA+S.	$\mathbf{-1.242 \pm .001}$	$-1.198 \pm .003$	$-1.169 \pm .003$	$-1.151 \pm .002$	$-1.131 \pm .003$
			$z = 2$		
NOPA NL	$-.999 \pm .001$	$-.995 \pm .001$	$-.981 \pm .001$	$-.980 \pm .001$	$-1.065 \pm .001$
NOPA+S. NL	$-.999 \pm .001$	$-.979 \pm .001$	$-.973 \pm .001$	$-.987 \pm .001$	$-1.064 \pm .001$
NOPA	$-1.174 \pm .001$	$-1.135 \pm .001$	$-1.119 \pm .001$	$-1.101 \pm .001$	$-1.083 \pm .001$
NOPA+S.	$-1.152 \pm .002$	$-1.130 \pm .001$	$-1.103 \pm .001$	$-1.080 \pm .001$	$-1.061 \pm .001$
INOPA	$\mathbf{-1.234 \pm .001}$	$\mathbf{-1.251 \pm .001}$	$\mathbf{-1.237 \pm .001}$	$\mathbf{-1.219 \pm .001}$	$\mathbf{-1.197 \pm .001}$
INOPA+S.	$-1.085 \pm .001$	$-1.085 \pm .001$	$-1.083 \pm .001$	$-1.083 \pm .001$	$-1.085 \pm .001$

- validate the use of portfolio (almost as good as the best solver, and sometimes better thanks to its inherent mitigation of "bad luck runs"); we incidentally provide, with INOPA applied to several independent copies of a same solver, a principled tool for restarts for noisy optimization;
- validate the improvement provided by unfair budget, as shown by the improvement of INOPA vs NOPA (when no sharing is applied, i.e. in the context in which our mathematical results are proved) - more precisely, we get either very similar results (in Table 3 and for $z = 0$ or $z = 2$ in Table 1, INOPA and NOPA

have essentially the same behavior), or a consistent improvement of INOPA vs NOPA ($z = 1$ in Table 1 and Tables 2, 4);
- are less conclusive in terms of comparison "with *lag* / without *lag*", though on average *lag* is seemingly beneficial.

4 Conclusion

We have seen that noisy optimization provides a very natural framework for portfolio methods. Different noisy optimization algorithms have extremely different convergence rates (different slopes) on different test cases, depending on the noise level, on the multimodalities, on the dimension (see e.g. Tables 1 and 4, where depending on z the best solver is a variant of Fabian or Newton's algorithm; and Table 3, where RSAES is the best); see also [29] for experiments on additional multimodal test cases. We proposed two versions of such portfolios, NOPA and INOPA, the latter using an unfair distribution of the budget. Both have theoretically the same slope as the best of their solvers, with better constants for INOPA (in particular, no shift, if $SubSetOptim$ (see Eq. 8) has cardinal 1).

We show mathematically an asymptotic $\log(M)$ shift when using M solvers, when working on a classical log-log scale (classical in noisy optimization); see Section 2.4.2. Contrarily to noise-free optimization (where a $\log(M)$ shift would be a trivial result), such a shift is not so easily obtained in noisy optimization. Importantly, it is necessary (Section 2.4.4), for getting the $\log(M)$ shift, that:

- the AS algorithm compares *old* recommendations (and selects a solver from this point of view);
- the portfolio recommends the *current* recommendation of this selected solver.

Additionally, we improve the bound to a $\log(M')$ shift, where M' is the number of optimal solvers, using an unfair distribution of the computational budget (Section 2.4.3). In particular, the shift is asymptotically negligible when the optimal solver is unique.

A careful choice of portfolio parameters (function $\textsc{lag}(\cdot)$, specifying the *lag*; r_n, specifying the intervals $r_{n+1} - r_n$ between two comparisons of solvers; s_n, specifying the number of resamplings of recommendations for selecting the best) leads to such properties; we provide principled tools for choosing these parameters. Sufficient conditions are given in Theorem 1, with examples thereafter.

Experiments show (i) the efficiency of portfolios for noisy optimization, as solvers have very different performances for different test cases and NOPA has performance close to the best or even better when the random initialization has a big impact; (ii) the clear and stable improvement provided by INOPA, thanks to an unfair budget distribution; (iii) that the *lag* is usually beneficial, though this is not always the case. Importantly, without lag, INOPA could not be defined.

In noisy frameworks, we point out that portfolios might make sense even when optimizers are not orthogonal. Even with identical solvers, or closely related optimizers, the portfolio can mitigate the effect of unlucky random contributions. This is somehow related to restarts (i.e. multiple runs with random initializations). See Table 4 for cases with very close solvers, and [29] with identical solvers.

Sharing information in portfolios of noisy optimization algorithms is not so easy. Our empirical results are mitigated; but we only tested very simple tools for

sharing - just sharing the current best point. A further work consists in identifying better relevant information for sharing; maybe the estimate of the asymptotic fitness value of a solver is the most natural information for sharing; if a fitness value A is already found and a solver claims that it will never do better than A, then we can safely stop its run and save up computational power.

References

1. Aha, D.W.: Generalizing from case studies: A case study. In: Proceedings of the 9th International Workshop on Machine Learning. pp. 1–10. Morgan Kaufmann Publishers Inc. (1992)
2. Armstrong, W., Christen, P., McCreath, E., Rendell, A.P.: Dynamic algorithm selection using reinforcement learning. In: International Workshop on Integrating AI and Data Mining. pp. 18–25 (2006)
3. Arnold, D.V., Beyer, H.G.: A general noise model and its effects on evolution strategy performance. IEEE Transactions on Evolutionary Computation 10(4), 380–391 (2006)
4. Astete-Morales, S., Liu, J., Teytaud, O.: Log-log Convergence for Noisy Optimization. In: Evolutionary Algorithms 2013. pp. 16 – 28. Proceedings of EA 2013, Bordeaux, France (2013), https://hal.inria.fr/hal-01107772
5. Astete-Morales, S., Cauwet, M.L., Liu, J., Teytaud, O.: Simple and cumulative regret for continuous noisy optimization. Theoretical Computer Science p. Accepted (2015)
6. Auer, P.: Using confidence bounds for exploitation-exploration trade-offs. The Journal of Machine Learning Research 3, 397–422 (2003)
7. Auer, P., Cesa-Bianchi, N., Freund, Y., Schapire, R.E.: Gambling in a rigged casino: the adversarial multi-armed bandit problem. In: Proceedings of the 36th Annual Symposium on Foundations of Computer Science. pp. 322–331. IEEE Computer Society Press, Los Alamitos, CA (1995)
8. Beyer, H.G.: The Theory of Evolutions Strategies. Springer, Heidelberg (2001)
9. Beyer, H.G.: Actuator noise in recombinant evolution strategies on general quadratic fitness models. In: Deb, K. (ed.) Genetic and Evolutionary Computation, GECCO 2004, Lecture Notes in Computer Science, vol. 3102, pp. 654–665. Springer Berlin Heidelberg (2004)
10. Borrett, J., Tsang, E.P.K.: Towards a formal framework for comparing constraint satisfaction problem formulations. Tech. rep., University of Essex, Department of Computer Science (1996)
11. Bubeck, S., Munos, R., Stoltz, G.: Pure exploration in multi-armed bandits problems. In: ALT. pp. 23–37 (2009)
12. Cauwet, M.L., Liu, J., Teytaud, O.: Algorithm Portfolios for Noisy Optimization: Compare Solvers Early. In: Learning and Intelligent Optimization Conference. Florida, USA (Mar 2014), http://hal.inria.fr/hal-00926638
13. Chen, H.: Lower rate of convergence for locating the maximum of a function. Annals of statistics 16, 1330–1334 (Sep 1988)
14. Conn, A., Scheinberg, K., Toint, P.: Recent progress in unconstrained nonlinear optimization without derivatives. Mathematical Programming 79(1-3), 397–414 (1997), http://dx.doi.org/10.1007/BF02614326
15. Couetoux, A.: Monte Carlo Tree Search for Continuous and Stochastic Sequential Decision Making Problems. Theses, Université Paris Sud - Paris XI (Sep 2013), https://tel.archives-ouvertes.fr/tel-00927252
16. Coulom, R.: Clop: Confident local optimization for noisy black-box parameter tuning. In: Advances in Computer Games, pp. 146–157. Springer Berlin Heidelberg (2012)
17. Fabian, V.: Stochastic approximation. In: Rustagi, J. (ed.) Optimization methods in Statistics; proceedings Symposium. pp. 439–470. Ohio State University, Academic Press, New York (1971)
18. Fabian, V.: Stochastic Approximation of Minima with Improved Asymptotic Speed. Annals of Mathematical statistics 38, 191–200 (1967)
19. Gagliolo, M., Schmidhuber, J.: A neural network model for inter-problem adaptive online time allocation. In: 15th International Conference on Artificial Neural Networks: Formal Models and Their Applications. pp. 7–12. Springer (2005)

20. Gagliolo, M., Schmidhuber, J.: Learning dynamic algorithm portfolios. Ann. Math. Artif. Intell. 47(3-4), 295–328 (2006)
21. Grigoriadis, M.D., Khachiyan, L.G.: A sublinear-time randomized approximation algorithm for matrix games. Operations Research Letters 18(2), 53–58 (Sep 1995)
22. Hamadi, Y.: Combinatorial Search: From Algorithms to Systems. Springer (2013)
23. Jebalia, M., Auger, A., Hansen, N.: Log linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. Algorithmica (2010), http://hal.inria.fr/inria-00433347
24. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments. a survey. IEEE Transactions on Evolutionary Computation 9(3), 303–317 (2005)
25. Kadioglu, S., Malitsky, Y., Sabharwal, A., Samulowitz, H., Sellmann, M.: Algorithm selection and scheduling. In: 17th International Conference on Principles and Practice of Constraint Programming. pp. 454–469 (2011)
26. Kocsis, L., Szepesvari, C.: Discounted-UCB. In: 2nd Pascal-Challenge Workshop (2006)
27. Kotthoff, L.: Algorithm selection for combinatorial search problems: A survey. CoRR abs/1210.7959 (2012)
28. Lai, T., Robbins, H.: Asymptotically efficient adaptive allocation rules. Advances in Applied Mathematics 6, 4–22 (1985)
29. Liu, J., Teytaud, O.: Meta online learning: experiments on a unit commitment problem. In: European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning (2014)
30. Prügel-Bennett, A.: Benefits of a population: Five mechanisms that advantage population-based algorithms. IEEE Trans. Evolutionary Computation 14(4), 500–517 (2010), http://dx.doi.org/10.1109/TEVC.2009.2039139
31. Pulina, L., Tacchella, A.: A self-adaptive multi-engine solver for quantified boolean formulas. Constraints 14(1), 80–116 (2009)
32. Rolet, P., Teytaud, O.: Adaptive Noisy Optimization. In: EvoStar 2010. pp. 592–601. Istambul, Turquie (Feb 2010), http://hal.inria.fr/inria-00459017
33. Samulowitz, H., Memisevic, R.: Learning to solve qbf. In: Proceedings of the 22nd National Conference on Artificial Intelligence. pp. 255–260. AAAI (2007)
34. Sendhoff, B., Beyer, H.G., Olhofer, M.: The influence of stochastic quality functions on evolutionary search. Recent Advances in Simulated Evolution and Learning, ser. Advances in Natural Computation pp. 152–172 (2004)
35. Shamir, O.: On the complexity of bandit linear optimization. In: Proceedings of The 28th Conference on Learning Theory, COLT 2015, Paris, France, July 3-6, 2015. pp. 1523–1551 (2015), http://jmlr.org/proceedings/papers/v40/Shamir15.html
36. Spall, J.: Adaptive stochastic approximation by the simultaneous perturbation method. Automatic Control, IEEE Transactions on 45(10), 1839–1853 (Oct 2000)
37. Spall, J.: Feedback and weighting mechanisms for improving jacobian estimates in the adaptive simultaneous perturbation algorithm. Automatic Control, IEEE Transactions on 54(6), 1216–1229 (June 2009)
38. St-Pierre, D.L., Liu, J.: Differential Evolution Algorithm Applied to Non-Stationary Bandit Problem. In: IEEE Congress on Evolutionary Computation (IEEE CEC). Beijing, China (Jul 2014), http://hal.inria.fr/hal-00979456
39. Storn, R., Price, K.: Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces. J. of Global Optimization 11(4), 341–359 (Dec 1997), http://dx.doi.org/10.1023/A:1008202821328
40. Sutton, R.S., Barto, A.G.: Reinforcement learning: An introduction. MIT Press., Cambridge, MA (1998)
41. Utgoff, P.E.: Perceptron trees: A case study in hybrid concept representations. In: National Conference on Artificial Intelligence. pp. 601–606 (1988)
42. Vassilevska, V., Williams, R., Woo, S.L.M.: Confronting hardness using a hybrid approach. In: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm. pp. 1–10. ACM (2006)
43. Weinstein, A., Littman, M.L.: Bandit-based planning and learning in continuous-action markov decision processes. In: Proceedings of the Twenty-Second International Conference on Automated Planning and Scheduling, ICAPS 2012, Atibaia, São Paulo, Brazil, June 25-19, 2012 (2012), http://www.aaai.org/ocs/index.php/ICAPS/ICAPS12/paper/view/4697
44. Wolpert, D.H., Macready, W.G.: No free lunch theorems for optimization. IEEE Transactions on Evolutionary Computation 1(1), 67–82 (Apr 1997)

45. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: Hydra-mip: automated algorithm configuration and selection for mixed integer programming. In: RCRA Workshop on Experimental Evaluation of Algorithms for Solving Problems with Combinatorial Explosion at the International Joint Conference on Artificial Intelligence (IJCAI) (2011)
46. Xu, L., Hutter, F., Hoos, H.H., Leyton-Brown, K.: SATzilla: Portfolio-based algorithm selection for SAT. J. Artif. Intell. Res. (JAIR) 32, 565–606 (2008), http://dx.doi.org/10.1613/jair.2490

A Appendix: Noisy optimization algorithms

We present briefly several noisy optimization algorithms. Algorithm 4 is a classical Self Adaptive-(μ,λ)-Evolution Strategy, with noise handled by resamplings. Algorithm 5 is a stochastic gradient method, with gradient estimated by finite differences; it is known to converge with simple regret $O(1/n)$ on smooth enough functions corrupted by additive noise [18, 35]. Algorithm 6 extends Fabian's algorithm by adding second-order information, by approximating the Hessian [17].

Algorithm 4 Self-adaptive Evolution Strategy with resamplings. $\mathcal{N}$ denotes some independent standard Gaussian random variable and $c = mod(a, b)$ for $b > 0$ means $\exists k \in \mathbb{Z}, (a - c) = bk$ and $0 \leq c < b$.

Require: dimension $d \in \mathbb{N}^*$
Require: population size $\lambda \in \mathbb{N}^*$ and number of parents $\mu \in \mathbb{N}^*$ with $\lambda \geq \mu$
Require: $K > 0$ ▷ parameter used to compute resampling number
Require: $\zeta \geq 0$ ▷ parameter used to compute resampling number
Require: an initial parent $x_{1,i} \in \mathbb{R}^d$ and an initial $\sigma_{1,i} = 1$, $i \in \{1, \dots, \mu\}$
$n \leftarrow 1$
$\tilde{x} \leftarrow x_{1,1}$ ▷ recommendation
while (true) **do**
  Generate λ individuals i_j, $j \in \{1, \dots, \lambda\}$, independently using ▷ offspring

$$\sigma_j = \sigma_{n,mod(j-1,\mu)+1} \times \exp\left(\frac{\mathcal{N}}{2d}\right) \text{ and } i_j = x_{n,mod(j-1,\mu)+1} + \sigma_j \mathcal{N}$$

  Evaluate each of them $\lceil Kn^\zeta \rceil$ times and average their fitness values
  Define $j_1, \dots, j_\lambda$ so that ▷ ranking

$$\hat{\mathbb{E}}_{\lceil Kn^\zeta \rceil}[f(i_{j_1})] \leq \hat{\mathbb{E}}_{\lceil Kn^\zeta \rceil}[f(i_{j_2})] \cdots \leq \hat{\mathbb{E}}_{\lceil Kn^\zeta \rceil}[f(i_{j_\lambda})]$$

  where $\hat{\mathbb{E}}_m$ denotes the average over m resamplings
  Compute $x_{n+1,k}$ and $\sigma_{n+1,k}$ using ▷ update

$$\sigma_{n+1,k} = \sigma_{j_k} \text{ and } x_{n+1,k} = i_{j_k},\ k \in \{1, \dots, \mu\}$$

  $\tilde{x} = i_{j_1}$ ▷ update recommendation
  $n \leftarrow n + 1$
end while

Algorithm 5 Fabian's stochastic gradient algorithm with finite differences. Fabian, in [18], proposes various rules for the parametrization; in the present paper, we use the following parameters. s is as in Remark 5.2 in [18], i.e., s is the minimal even number $\geq \frac{1}{2\gamma} - 1$. The scales u_i are $u_i = \frac{1}{i}, \forall i \in \{1, \ldots, \frac{s}{2}\}$; this generalizes the choice in Example 3.3 in [18]. The w_i are computed as given in Lemma 3.1 in [18]. e_i is the i^{th} vector of the standard orthonormal basis of $\mathbb{R}^d$.

Require: dimension $d \in \mathbb{N}^*$
Require: $\frac{1}{2} > \gamma > 0$, $a > 0$, $c > 0$, even number of samples per axis s
Require: scales $1 \geq u_1 > \cdots > u_{\frac{s}{2}} > 0$, weights $w_1 > \cdots > w_{\frac{s}{2}}$ summing to 1
Require: an initial $x_1 \in \mathbb{R}^d$
$n \leftarrow 1$
$\tilde{x} \leftarrow x_1$ ▷ recommendation
while (true) **do**
  Compute $\sigma_n = c/n^\gamma$ ▷ step-size
  Evaluate the gradient g at x_n by finite differences, averaging over s samples per axis: $\forall i \in \{1, \ldots, d\}, \forall j \in \{1, \ldots, \frac{s}{2}\}$ ▷ gradient estimation

$$x_n^{(i,j)+} = x_n + u_j \sigma_n e_i \quad \text{and} \quad x_n^{(i,j)-} = x_n - u_j \sigma_n e_i$$
$$g_i = \frac{1}{2\sigma_n} \sum_{j=1}^{s/2} w_j \left(f(x_n^{(i,j)+}) - f(x_n^{(i,j)-}) \right)$$

  Apply $x_{n+1} = x_n - \frac{a}{n} g$ ▷ next search point
  $\tilde{x} \leftarrow x_{n+1}$ ▷ update recommendation
  $n \leftarrow n + 1$
end while

B Summary of notations

Notations are as follows:

General notations:

$\mathbb{E}_\omega$ = expectation with respect to random variable ω.
$\hat{\mathbb{E}}_k X$ = average over k independent realizations of random variable X.

Notation for solvers:

x_n = search point used by the solver for the n^{th} evaluation.
$\tilde{x}_n$ = recommendation given by the solver after the n^{th} evaluation.
$SR_n = \mathbb{E}\left(f(\tilde{x}_n) - f(x^*)\right)$. (simple regret)

Notation for AS algorithms:

i^* = index of the solver chosen by the AS algorithm.
$\tilde{x}_{i,n}$ = recommendation given by the solver i after the n^{th} evaluation.
$SR_{i,n} = \mathbb{E}\left(f(\tilde{x}_{i,n}) - f(x^*)\right)$.
M = number of solvers in portfolio.
$\Delta_{i,n} = SR_{i,n} - \min_{j \in \{1,\ldots,M\}} SR_{j,n}$.
$SR_n^{Solvers} = \min_{i \in \{1,\ldots M\}} SR_{i,n}$.
$SR_n^{Selection} = \mathbb{E}\left(f(\tilde{x}_{i^*,n}) - f(x^*)\right)$.

Algorithm 6 An adaptation of Newton's algorithm for noisy objective functions, with gradient and Hessian approximated by finite differences and reevaluations. The recommendations are the x_n's. e_i is the i^{th} vector of the standard orthonormal basis of $\mathbb{R}^d$.

Require: dimension $d \in \mathbb{N}^*$
Require: $A > 0$, $B > 0$, $\alpha > 0$, $\beta > 0$
Require: an initial $x_1 \in \mathbb{R}^d$
$n \leftarrow 1$
$\tilde{x} \leftarrow x_1$ ▷ recommendation
$\hat{h} \leftarrow$ identity matrix
while (true) **do**
  Compute $\sigma_n = A/n^\alpha$ ▷ step-size
  for $i = 1$ to d **do**
    Evaluate g_i by finite differences at $x_n + \sigma_n e_i$ and $x_n - \sigma_n e_i$, averaging each evaluation over $\lceil Bn^\beta \rceil$ resamplings.
  end for
  for $i = 1$ to d **do**
    Evaluate $\hat{h}_{i,i}$ by finite differences at $x_n + \sigma_n e_i$, x_n and $x_n - \sigma_n e_i$, averaging each evaluation over $\lceil Bn^\beta \rceil$ resamplings
    for $j = 1$ to d, $j \neq i$ **do**
      Evaluate $\hat{h}_{i,j}$ by finite differences thanks to evaluations at each of $x_n \pm \sigma_n e_i \pm \sigma_n e_j$, averaging over $\lceil Bn^\beta/10 \rceil$ resamplings
    end for
  end for
  $\delta \leftarrow$ solution of $\hat{h}\delta = -g$ ▷ possible next search point
  if $\|\delta\| > \frac{1}{2}\sigma_n$ **then**
    $\delta = \frac{1}{2}\sigma_n \frac{\delta}{\|\delta\|}$ ▷ trust region style
  end if
  Apply $x_{n+1} = x_n + \delta$
  $\tilde{x} \leftarrow x_{n+1}$ ▷ update recommendation
  $n \leftarrow n + 1$
end while