A Simple Yet Effective Resampling Rule in Noisy Evolutionary Optimization

Jialin Liu
Shenzhen Key Laboratory of Computational Intelligence
University Key Laboratory of Evolving Intelligent Systems of Guangdong Province
Department of Computer Science and Engineering
Southern University of Science and Technology
Shenzhen 518055, China
Email: liujl@sustech.edu.cn

Olivier Teytaud
Inria TAU, LRI, UMR 8623
CNRS - Univ. Paris-Saclay
Gif-sur-Yvette 91190, France
Email: olivier.teytaud@inria.fr

***Abstract*—Noisy optimization refers to the optimization of objective functions corrupted by noise, which happens in many real-world optimization problems. Resampling has been widely used in evolutionary algorithms for noisy optimization. It has been theoretically proved that evolutionary algorithms with resampling can achieve a "log-log convergence" slope of $-\frac{1}{2}$ when optimizing functions corrupted by unbiased additive noise [1]. Various dynamic resampling rules have been proposed in the literature. However, determining their optimal hyperparameter values for reaching the optimal slope is hard. In this work, we reach this slope using resampling rules optimized numerically though automatic parameter tuning. We have found a parameter-free yet effective new resampling rule depending on the iteration number and the problem dimension. This simple parameter-free resampling rule is compared to several state-of-the-art rules and achieved superior performance on functions corrupted by asymmetric additive noise or in case of very high noise levels.**

***Index Terms*—Noisy optimization, resampling rule, additive noise, evolution strategies, automatic parameter tuning**

I. Introduction

Resampling, i.e., explicitly averaging multiple evaluations of an identical solution, have been widely used in evolutionary algorithms for solving noisy optimization problems [2], [3]. Various dynamic resampling rules, some of which, with parameters have been studied in the literature and have been mathematically or empirically proved to be effective within different evolutionary algorithms for problems corrupted by constant variance noise and multiplicative noise [1], [3]–[7].

In this paper, we combine several resampling rules to a dynamic stopping rule, then apply automatic parameter tuning for optimizing the resampling rule of an evolutionary algorithm for noisy optimization. A parameter-free resampling rule is obtained by parameter tuning plus rounding of the obtained parameter values. The resulting resampling rule is simple but efficient on optimizing functions corrupted by additive noise of different strengths. Then, this rule is compared to several state-of-the-art algorithms on functions corrupted by symmetric and asymmetric noise. The comparison experiments are carried out using the open source Python3 library *nevergrad* [8] for derivative-free optimization. Our rule performs well in case of high noise or misleading noise models.

The paper is organized as follows. Section II introduces the framework including some related work in noisy optimization. Our proposed method is detailed in Section III. Section IV provides some experimental results on automatic design of resampling rules. Then, Section V presents the resulting approximate resampling rule and compares it to some state-of-the-art algorithms. Finally, Section VI concludes this paper.

This work was supported by National Key R&D Program of China (Grant No. 2017YFC0804003), National Natural Science Foundation of China (Grant No. 61906083), Shenzhen Peacock Plan (Grant No. KQTD2016112514355531), the Science and Technology Innovation Committee Foundation of Shenzhen (Grant No. ZDSYS201703031748284) and the Program for University Key Laboratory of Guangdong Province (Grant No. 2017KSYS008).

II. Background

A. Noisy black-box optimization in continuous domain

Noisy optimization is the optimization of objective functions corrupted by noise. A black-box noisy optimization consists in searching for the optimum (e.g. minimum) $\boldsymbol{x}^* = \text{argmin}_{\boldsymbol{x}\in\mathbb{R}^d} \mathbb{E}_{\boldsymbol{w}} f(\boldsymbol{x}, \boldsymbol{w})$ of some noisy objective function $f : \mathbb{R}^d \mapsto \mathbb{R}$ by successive calls to f without using any internal property of the objective function. f is a random process, and equivalently it can be viewed as a mapping $(\boldsymbol{x}, \boldsymbol{w}) \mapsto f'(\boldsymbol{x}, \boldsymbol{w})$ where $\boldsymbol{x} \in \mathbb{R}^d$ and $\boldsymbol{w}$ is a random variable independently sampled at each call to f. We aim at finding a good approximation of $\boldsymbol{x}^*$. Various "classical" algorithms have been proposed for noisy optimization in continuous domains. We here consider noisy evolutionary optimization.

a) Optimization criterion: Simple Regret (SR) is usually used as a criterion for estimating the performance of noisy optimization algorithms. $\tilde{\boldsymbol{x}}_m$ is the approximate optimum obtained by the optimization algorithm after m function evaluations. SR_m, defined as $SR_m = \mathbb{E}_{\boldsymbol{w}} f(\tilde{\boldsymbol{x}}_m) - \mathbb{E}_{\boldsymbol{w}} f(\boldsymbol{x}^*)$, is the SR after m evaluations. The slope of SR [1] is defined as:

$$slope(SR) = \limsup_m \frac{\log\left(\mathbb{E}_{\boldsymbol{w}} f(\tilde{\boldsymbol{x}}_m, \boldsymbol{w}) - \mathbb{E}_{\boldsymbol{w}} f(\boldsymbol{x}^*, \boldsymbol{w})\right)}{\log m}. \quad (1)$$

b) Noise models: Two main categories of noise have been proposed and investigated in the literature: *prior* and *posterior*. Prior noise refers to the perturbation of solution ($\boldsymbol{x}$) and happens before the evaluation process. Its noisy objective function can be expressed as $f(\boldsymbol{x}, \boldsymbol{w}) = f_t(\boldsymbol{x} + \boldsymbol{w})$, where $f_t(\cdot)$ denotes the true noise-free objective function. The prior noise is mostly considered in the discrete domain [9], [10], which is beyond the scope of this paper. Posterior noise refers to the perturbation of objective function, thus evaluation process. Various posterior noise models have been studied [2], [3]. The two main categories that have been considered mostly in the continuous domain and sometimes in the discrete domain are *multiplicative noise* of variance scaled with the objective function ("weakly noisy") [4], [5], [11]–[16], where $f(\boldsymbol{x}, \boldsymbol{w}) = f_t(\boldsymbol{x}) * \boldsymbol{w}$ or $f(\boldsymbol{x}, \boldsymbol{w}) = f_t(\boldsymbol{x}) * (1 + \boldsymbol{w})$, and *additive noise of constant variance* ("strongly noisy") [1], [4]–[7], where $f(\boldsymbol{x}, \boldsymbol{w}) = f_t(\boldsymbol{x}) + \boldsymbol{w}$. In this work, we focus on the strongly noisy case in the continuous domain, i.e., the variance of the noise does not decrease near the optimum, which requires more resampling to reduce the probability of misranking the solutions when searching around the optimum. The most popular model is the additive Gaussian noise [3].

B. Noise handling techniques

The two main methods for handling noise in evolutionary algorithms [17] are increasing the population size and resampling (i.e., averaging multiple evaluations of a solution $\boldsymbol{x}$).

[18] presented a technique called "mutate large, but inherit small" (MLIS), which improved the convergence of $(1, \lambda)$-ES. The approximation quality, i.e. the expected Euclidean distance change toward the optimum, increased with larger population and decreased with increasing mutation strength (step-size). Hence, MLIS proposes to use a shrink-rescaled parental step-size to handle the problem.

It has been theoretically proved that evolutionary algorithms with resampling can achieve a "log-log convergence" slope (thus $slope(SR)$) of $-\frac{1}{2}$ when optimizing functions corrupted by unbiased additive noise [1]. This means that the logarithm of the simple regret, i.e., the difference between the evaluation of current recommendation and the true optimal value, converges linearly in proportion $-\frac{1}{2}$ of the logarithm of the number of evaluations. Astete et al. [19] have proved that, when *simple* evolutionary strategies (simple-ES) optimize sphere functions corrupted by additive Gaussian noise of constant variance, the $slope(SR)$ is lower bounded by $-\frac{1}{2}$.

Recently, Friedrich et al. [20] proved theoretically that when the noise is strong enough, increasing the population size does not help. When the noise is weak, resampling is not necessary. Therefore, in this work, we consider approaches based on resampling in the strongly noisy case.

Here, we consider three families of resampling methods: (i) resampling based on the iteration number [1], [6]; (ii) resampling based on the step-size [1]; and (iii) resampling based on statistical testing [21]–[23]. The formulation of these resampling strategies are given in Section III. All these resampling methods are controlled by one or more parameters, which affect significantly the convergence rate of noisy optimization algorithms. The determination of the optimal parameter values is not trivial. We aim at finding a parameter-free but efficient resampling rule. We combine the above resampling methods into one single formula, then tune the parameters automatically by optimising the convergence rate obtained using the resampling determined by the formula with tuned parameters (Sections III and IV). A new, simple yet effective resampling rule is found as detailed in Section V.

Algorithm 1 A modified (1+1)-ES with resampling for a minimization problem. $\mathcal{N}^d(0,1)$ denotes a standard d-dimensional Gaussian noise. In this algorithm, $\boldsymbol{x}_n$ refers to the parent at the n^{th} iteration.

Require: $d \in \mathbb{N}^+$ ▷ dimension of search space
Require: a noisy fitness function $f : \mathbb{R}^d \mapsto \mathbb{R}$
Require: a resampling rule $r(\cdot) : \cdot \mapsto \mathbb{N}^+$
Require: $\sigma_0 > 0$ ▷ initial mutation strength (step-size)
Require: $\boldsymbol{x}_0 \in \mathbb{R}^d$ ▷ initial individual
$n \leftarrow 0$ ▷ iteration number
$m \leftarrow 0$ ▷ evaluation number
$evals_p \leftarrow 0$ ▷ parent's evaluation number
$y_p \leftarrow 0$ ▷ parent's empirical evaluation
while budget is not exhausted **do**
  $\boldsymbol{x}' \leftarrow \boldsymbol{x}_n + \sigma \times \mathcal{N}^d(0,1)$ ▷ generate offspring
  $r_n \leftarrow r(\cdot)$ ▷ compute resampling number
  $y \leftarrow \overline{\mathbb{E}}_{r_n} f(\boldsymbol{x}_n)$ ▷ evaluate parent r times
  $y' \leftarrow \overline{\mathbb{E}}_{r_n} f(\boldsymbol{x}')$ ▷ evaluate offspring r times
  $y \leftarrow (y_p \cdot evals_p + y \cdot r_n)/(evals_p + r_n)$
  if $y' < y$ **then** ▷ offspring is better
    $\boldsymbol{x}_{n+1} \leftarrow \boldsymbol{x}'$
    $\sigma_{n+1} \leftarrow 2\sigma_n$
    $y_p \leftarrow y'$
    $evals_p \leftarrow r_n$
  else ▷ offspring is not better
    $\sigma_{n+1} \leftarrow 0.84\sigma_n$
    $y_p \leftarrow y$
    $evals_p \leftarrow evals_p + r_n$
  end if
  $n \leftarrow n + 1$ ▷ increment iteration number
  $m \leftarrow m + 2r_n$ ▷ update evaluation number
end while

III. Proposed Method

Section III-A introduces some notations. Section III-B describes our modified (1+1)-Evolution Strategy with resampling. Section III-C and Section III-D define some non-adaptive and adaptive resampling rules, as well as their combined formulation. Then we propose our dynamic stopping rule in Section III-E.

A. Notations and noise model

In this paper, $\mathbb{N}^+ = \{1, 2, 3, \dots\}$. If X is a random variable, then $X^{(1)}, X^{(2)}, \dots$ denotes samples of independent identically distributed random variables, copies of X. To simplify

the notation, from now on, $f(\boldsymbol{x})$ refers to an independent call to $f(\boldsymbol{x}, \boldsymbol{w})$, the operator $\mathbb{E}$ refers to $\mathbb{E}_{\boldsymbol{w}}$. We denote by $\overline{\mathbb{E}}_r f(\boldsymbol{x})$ the empirical evaluation of $\mathbb{E}f(\boldsymbol{x})$ over $r \in \mathbb{N}^+$ resamplings.

In this paper, we focus on studying unbiased additive noise, i.e., constant variance noise and independent to the noise-free fitness value. A commonly studied noise model is Gaussian noise, where the corresponding noisy objective function is defined as $f(\boldsymbol{x}, \boldsymbol{w}) = f(\boldsymbol{x}) + \varphi \times \mathcal{N}^d(0, 1)$, where $\mathcal{N}^d(0, 1)$ denotes a standard d-dimensional Gaussian distribution and φ refers to the noise strength, i.e. given $\boldsymbol{x} \in \mathbb{R}^d$, $\mathcal{V}arf(x) = \varphi^2$. Thus, the noise has a constant variance and is independent of $\boldsymbol{x}$ and its corresponding noise-free objective value. The noise strength does not vanish around the optimum.

B. (1+1)-Evolution Strategy with resampling

We use a modified (1+1)-Evolution Strategy with *resampling* as shown in Algorithm 1. The core difference is the computation of resampling number and averaging of evaluations of solution points (lines 7-10 of Algorithm 1).

C. Non-adaptive resampling rule

Astete et al. [1] have shown that the logarithm of the distance to the optimum converges linearly in the logarithm of the number of evaluations when using non-adaptive resampling rules which are exponential or polynomial in the iteration number n or polynomial in n. Liu et al. [6] have further compared eight non-adaptive sampling rules based on the iteration number and/or the dimension of the problem. The studied non-adaptive resampling rules in [1], [6] can be summarized into two formulas: $A_n = \lceil \zeta^{\frac{n}{d^\kappa}} \rceil$ and $B_n = \lceil (\frac{n}{d^\kappa})^\zeta \rceil$, where $\kappa \in [0, \infty)$ and $\zeta \in (0, \infty)$ are two control parameters. n is the current iteration number and d is the dimension of problem. Using these two formulas, we define the following rule:

$$r_n = \lceil A_n^\varrho \times B_n^{1-\varrho} \rceil, \tag{2}$$

where $\varrho \in [0, 1]$ is an additional parameter. ϱ determines how to combine these two terms in resampling.

D. Adaptive resampling rule

Astete et al. [1] have proved mathematically that the log-log convergence exists when using adaptive resampling rules with the number of resampling polynomial in the reciprocal of the step-size at the n^{th} iteration, σ_n. We include this adaptive resampling rule from [1] and use its variant as:

$$C_n = \lceil (\frac{n}{d^\kappa})^\zeta {\sigma_n}^{-\eta} \rceil, \tag{3}$$

where $\eta \in [0, \infty)$. This resampling rule contains B_n when $\eta = 0$. So we can modify (2) and obtain a new formula:

$$r'_n = \lceil A_n^\varrho \times C_n^{1-\varrho} \rceil. \tag{4}$$

E. Stopping rule by Empirical Bernstein Bound

Sometimes, when lucky enough, there is no need to use up the computed resampling number of evaluations to correctly compare two solutions. We now propose an additional term, for possibly interrupting the increase of resampling number by Empirical Bernstein Bound [24], [25]. We first recall in Theorem 1, the stopping rule *EBStop*, provided by [24].

Theorem 1 (Empirical Bernstein Bound [24])**.** *For some real-valued* i.i.d. *random variables $X_1, X_2, \ldots, X_r$ with mean μ and $\forall i \in \{1, \ldots, r\}, |X_i| \leq M$, there exists some positive sequence $(\varepsilon_r)_{r \in \mathbb{N}^+}$ such that the event $\mathcal{E} = \{|\overline{\mathbb{E}}_r X - \mu| \leq \varepsilon_r, r \in \mathbb{N}^+\}$ occurs with probability at least $1 - \delta$, where $\overline{\mathbb{E}}_r X$ denotes $\frac{1}{r}\sum_{i=1}^r X_i$.*

Let X_i be the difference of the i^{th} call of $f(\boldsymbol{x})$ and the i^{th} call of $f(\boldsymbol{x}')$, $f(\boldsymbol{x}') - f(\boldsymbol{x})$, where $\boldsymbol{x}$ and $\boldsymbol{x}'$ are two independent points. By Theorem 1, for all $r \in \mathbb{N}^+$, there exists some positive sequence (ε_r) such as (5) such that

$$\mathbb{P}\left(\left|(\overline{\mathbb{E}}_r f(\boldsymbol{x}) - \overline{\mathbb{E}}_r f(\boldsymbol{x}')) - (\mathbb{E}f(\boldsymbol{x}) - \mathbb{E}f(\boldsymbol{x}'))\right| \leq \varepsilon_r\right) \geq 1 - \delta.$$

Example 1. *The following sequence $(\varepsilon_r)_{r \in \mathbb{N}^+}$ provided by [24] respects Theorem 1:*

$$\varepsilon_r = \overline{\theta}_r \sqrt{\frac{2\log(3/d_r)}{r}} + \frac{3M\log(3/d_r)}{r}, \tag{5}$$

where $d_r = c/r^p$, $c = \delta(p-1)/p$, p is some real number, and

$$\overline{\theta}_r^2 = \frac{1}{r}\sum_{i=1}^r \left(X_i - \overline{\mathbb{E}}_r X\right)^2. \tag{6}$$

Minh et al. [25] have set d_r as $\frac{1}{r(r+1)}$, which does not include additional parameters. To extract the core parameters to be configured, by $d_r = \frac{1}{r(r+1)}$, we simplify (5) to:

$$\varepsilon_r \leftarrow \alpha\overline{\theta}_r \sqrt{\frac{\log r}{r}} + \beta\frac{\log r}{r}, \tag{7}$$

where r is current resampling number, α and β are two parameters to be optimized with $\alpha, \beta > 1$. Using (7) and Example 1, we propose a new adaptive stopping rule using Empirical Bernstein Bound as detailed in Algorithm 2.

IV. Automatic design of resampling rules

Section IV-A summarizes the studied resampling rules. Section IV-B presents our meta-optimizer for optimizing resampling rules' parameters. Section IV-C describes the test cases. In Section IV-D we present and discuss the experimental results.

A. Resampling rules

We apply 3 adaptive and non-adaptive resampling rules, combinations of these 3 rules and their variants with stopping rule by Empirical Bernstein Bound, *EBStop*, presented in the previous section. Instead of doing one resampling at a time (line 6-8 in Algorithm 2), we use, in our experiments, an exponentially increasing number of resampling per block to accelerate the implementation as detailed in Application 1.

Algorithm 2 *EBStop*, stopping rule by Empirical Bernstein Bound. $f(\boldsymbol{x})^{(1)}, \ldots, f(\boldsymbol{x})^{(r)}$ denote the evaluations of $\boldsymbol{x}$ ($\boldsymbol{x}'$ *respectively*).

Require: ϵ: precision
Require: $x, x' \in \mathbb{R}^d$: two points to be compared
1: $LB \leftarrow \infty$
2: $UB \leftarrow -\infty$
3: $r \leftarrow 0$ ▷ Number of resampling
4: **while** $(1+\epsilon)LB < (1-\epsilon)UB$ **do**
5: $X^{(r+1)} \leftarrow f(\boldsymbol{x})^{(i)} - f(\boldsymbol{x}')^{(i)}$ ▷ Resample the points
6: $r \leftarrow r+1$ ▷ Update the number of resampling
7: Compute $\overline{\mathbb{E}}_r X$ ▷ Update the empirical mean
8: $\varepsilon_r \leftarrow \alpha\bar{\theta}_r\sqrt{\frac{\log r}{r}} + \beta\frac{\log r}{r}$
9: $LB \leftarrow \max\{LB, \overline{\mathbb{E}}_r X - \varepsilon_r\}$
10: $UB \leftarrow \min\{UB, \overline{\mathbb{E}}_r X + \varepsilon_r\}$
11: **end while**

We summarize in Table I these 6 resampling rules used in the experiments and their notations.

Application 1. *The resampling of parent and offspring (lines 7-10) in Algorithm 1 is replaced by the following lines.*

1: $r \leftarrow 0$ ▷ *Resampling counter*
2: $b \leftarrow 0$ ▷ *Resampling block index*
3: $y,\ y' \leftarrow 0$
4: ***while*** $(1+\epsilon)LB < (1-\epsilon)UB$ ***and*** $r < r_n$ ***do***
5: $r_b \leftarrow \min(10 \times 2^b, r_n - r)$ ▷ *Compute the block size*
6: ***for*** $i \in \{1, \cdots, r_b\}$ ***do***▷ *Resample the points r_b times*
7: $X^{(r+i)} \leftarrow f(\boldsymbol{x})^{(r+i)} - f(\boldsymbol{x}')^{(r+i)}$
8: ***end for***
9: $r \leftarrow r + r_b$
10: $y' \leftarrow \overline{\mathbb{E}}_r f(\boldsymbol{x}')$ ▷ *Update the empirical mean*
11: $y \leftarrow \overline{\mathbb{E}}_r f(\boldsymbol{x})$ ▷ *Update the empirical mean*
12: *Compute* $\overline{\mathbb{E}}_r X$ ▷ *Update the empirical mean*
13: *Compute* $\bar{\theta}_r$ *using* (6)
14: $\varepsilon_r \leftarrow \alpha\bar{\theta}_r\sqrt{\frac{\log r}{r}} + \beta\frac{\log r}{r}$
15: $LB \leftarrow \max\{LB, \overline{\mathbb{E}}_r X - \varepsilon_r\}$
16: $UB \leftarrow \min\{UB, \overline{\mathbb{E}}_r X + \varepsilon_r\}$
17: $b \leftarrow b + 1$ ▷ *Update block index*
18: ***end while***

TABLE I
ADAPTIVE AND NON-ADAPTIVE RESAMPLING RULES. τ_{EBB} REFERS TO THE ADAPTIVE STOPPING RULE BY EMPIRICAL BERNSTEIN BOUND, AS PRESENTED IN APPLICATION 1.

Rules	Formula	Parameter	Notation
Non-adaptive	(2)	ζ, κ, ϱ	r_{NA}
Non-adaptive with τ_{EBB}	(2) and (7)	ζ, κ, ϱ, α, β	$r_{NA,\tau_{EBB}}$
Adaptive	(3)	ζ, κ, η	r_{σ}
Adaptive with τ_{EBB}	(3) and (7)	ζ, κ, η, α, β	$r_{\sigma,\tau_{EBB}}$
Adaptive	(4)	ζ, κ, η, ϱ	$r_{NA,\sigma}$
Adaptive with τ_{EBB}	(4) and (7)	ζ, κ, η, ϱ, α, β	$r_{NA,\sigma,\tau_{EBB}}$

B. Meta-optimizer

We have two levels of optimization. The upper level is the design and tuning of a resampling rule. The lower level is a run of an evolution strategy, using the resampling rule provided by the upper level. We refer to the upper level as *meta-optimization*. The meta-optimization uses the convergence slope of the lower level as the objective value. We include a meta-optimizer which uses a $(1+1)$-ES to tune parameters of resampling rules defined in Table I. Our parameters are the ones deciding the number of resampling, i.e. $\zeta, \kappa, \eta, \varrho$ in (2), (3), (4) and α, β in (7). The search space of the parameters is as follows: ζ, κ and η range in $(0, \infty)$, α and β range in $(1, \infty)$, while ϱ takes real values between 0 and 1.

The approximate slope at iteration n [1] is defined as

$$s = \frac{\log \mathbb{E} f(\boldsymbol{x}_n)}{\log m}. \tag{8}$$

Again, $\boldsymbol{x}_n$ is the approximation of the optimum indexed by iteration number n and m is the total number of evaluations over iterations $1, 2, \ldots, n$. We will consider the first n such that a given number m of function evaluations is reached. Actually, we will interrupt the corresponding iteration. Therefore, $\tilde{\boldsymbol{x}}_m = \boldsymbol{x}_n$, $\tilde{\boldsymbol{x}}_m$ is the approximate optimum provided by the algorithm after m evaluations.

We define our objective function g as follows: s_i is the approximate slope in dimension 2^i ($i \in \{1 \ldots 6\}$), $g_1 = \frac{1}{6}\sum_{i=1}^{6} s_i$ is the average slope, $g_2 = \max_{i \in \{1 \ldots 6\}} s_i$ is the slope of the worst case, and the objective function is

$$g = \frac{1}{2}(g_1 + g_2). \tag{9}$$

The objective function g is a mapping from some function f to real number, i.e. $g : \mathcal{F} \mapsto \mathbb{R}$. For any $f \in \mathcal{F}$, $g(f)$ denotes the mean of, g_2, the worst approximate slope, and g_1, the averaged approximate slope in dimension $2^i, \forall i \in \{1, \ldots, 6\}$. The weights of g_1 and g_2 (set as $\frac{1}{2}$) are arbitrarily chosen.

Note that the meta-optimizer itself and the problem it aims to optimize are also noisy, we apply $(1+1)$-ES described in Algorithm 1 with resampling number 1.01^n at the n^{th} iteration. This simple exponential rule, denoted as *O101*, with a very small coefficient has lead to overall good performance on a set of benchmark functions in the strongly noisy case [7].

C. Testbed

We consider sphere functions with unbiased additive noise of different strengths and seven test cases, detailed in Table II. As the optimum of noisy sphere functions f_1, f_2 and f_3 are located in $\boldsymbol{x}^* = 0^d$. These three functions model the noise of different strength levels or of same level but starting from different initial points (near to far from the optimum). We get $\mathbb{E} f_1(\boldsymbol{x}^*) = \mathbb{E} f_2(\boldsymbol{x}^*) = \mathbb{E} f_3(\boldsymbol{x}^*) = 0$. By definition, the approximate slope defined in (8) is then the slope of SR.

For each test case, an iteration of meta-optimization is a tuning iteration of parameters. We first tune the parameters of all the resampling rules for each case (termed as *optimization phase*), then apply the optimized parameters, which provide better slopes, to all the 7 cases (termed as *test phase*).

TABLE II
TEST CASES AND RELATED OBJECTIVE FUNCTIONS. f_1, f_2 AND f_3 ARE 3 NOISY SPHERE FUNCTIONS. $\mathcal{N}$ DENOTES SOME INDEPENDENT STANDARD GAUSSIAN RANDOM VARIABLE.

Test case number	Objective function
$\mathcal{T}_1$	$g(f_1)$ with $f_1 = \|\|\boldsymbol{x}\|\|^2 + 1e^{-6}\mathcal{N}$
$\mathcal{T}_2$	$g(f_2)$ with $f_2 = \|\|\boldsymbol{x}\|\|^2 + 0.05\mathcal{N}$
$\mathcal{T}_3$	$g(f_3)$ with $f_3 = \|\|\boldsymbol{x}\|\|^2 + \mathcal{N}$
$\mathcal{T}_4$	$\frac{1}{2}(g(f_1) + g(f_2))$
$\mathcal{T}_5$	$\frac{1}{2}(g(f_1) + g(f_3))$
$\mathcal{T}_6$	$\frac{1}{2}(g(f_2) + g(f_3))$
$\mathcal{T}_7$	$\frac{1}{3}(g(f_1) + g(f_2) + g(f_3))$

D. Results and discussion

Experiments were performed with 600 iterations at the meta level and $5e5$ evaluations at the lower level (i.e., one evaluation of a given vector of parameter values). The initial point is $||\boldsymbol{x}_0|| = 1$ and the initial step-size is $\sigma_0 = 1$. The approximate slopes are presented in Table III.

We observe that: (i) The non-adaptive resampling rule r_{NA} and the adaptive rule $r_{NA,\sigma}$, performed well on all dimensions. (ii) However, the adaptive rule r_σ does not lead to convergence in the case with a high noise strength. (iii) Parameters optimized for the test case 1, corrupted by an additive very small constant variance noise, provide a slightly worse performance for cases with a big constant variance noise. (iv) Parameters optimized for the test case 3, corrupted by an additive big constant variance noise, do not provide good performance for cases with a small constant variance noise. Actually, for cases corrupted by a small constant variance noise do not need too many resamplings to cancel the effect of the noise.

V. OUR APPROXIMATE RESAMPLING RULE

A. Approximation of resampling rules

We summarize the overall best resampling rules, $r_{NA}^{\mathcal{T}_1}$ and $r_{NA}^{\mathcal{T}_4}$, obtained by optimizing on test case $\mathcal{T}_1$ and $\mathcal{T}_4$, respectively, but perform overall well on all test cases. These two rules are marked by "*" at the end of line in Table III, also shown as $r_{NA}^{\mathcal{T}_1} \approx \lceil 1.0751 \frac{n}{a^{1.0440}} \cdot \max\{1, \frac{n^{0.6362}}{d^{0.6642}}\}\rceil$ and $r_{NA}^{\mathcal{T}_4} \approx \lceil 1.0860 \frac{n}{d^{0.9990}} \cdot \max\{1, \frac{n^{0.3920}}{d^{0.3916}}\}\rceil$. We observe that the first term of $r_{NA}^{\mathcal{T}_1}$ increases slower than the first term of $r_{NA}^{\mathcal{T}_4}$, while its second term increases faster than the one of $r_{NA}^{\mathcal{T}_4}$. From $r_{NA}^{\mathcal{T}_1}$ and $r_{NA}^{\mathcal{T}_4}$, we propose the following approximate non-adaptive formula

$$\tilde{r}^* = \lceil 1.1^{\frac{n}{d}} \cdot \max\{1, \sqrt{\frac{n}{d}}\}\rceil. \tag{10}$$

We applied the approximate non-adaptive formula (10) to the first 3 test cases with weak, medium and strong noise strength using different budgets, in terms of the number of function evaluations. The initial point is $||\boldsymbol{x}_0|| = 1$ and the initial step-size is $\sigma_0 = 1$ as in the previously presented tuning phase. Table IV presents the averaged approximate slopes.

It can be seen that the approximate non-adaptive formula achieves slightly worse slopes to approximate slopes obtained by formulas with optimized parameters (presented in Table III). When using bigger budget, i.e. more function evaluations, we get much better slopes in high dimensions in the test case 3 (with bigger noise). The approximate non-adaptive formula is never so bad in all the three cases with different noise strength, i.e. it is more robust than other methods.

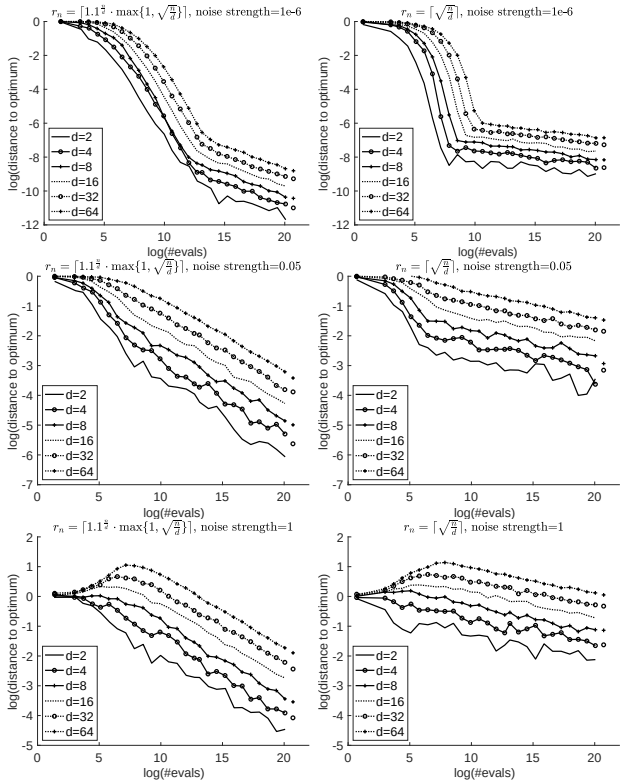

Fig. 1. The logarithm of the distance to the optimum converges linearly in the logarithm of the number of evaluations. Figures obtained using the approximate resampling rule $\tilde{r}^*$ (left) and the simplified rule $\tilde{r}$ (right).

B. The factor $1.1^{\frac{n}{d}}$ is important

We exam the effect of factor $1.1^{\frac{n}{d}}$ in the approximate rule by comparing it to an even simpler rule, formalized as:

$$\tilde{r} = \lceil \sqrt{\frac{n}{d}} \rceil. \tag{11}$$

The log-log convergence of both resampling rules on optimizing functions corrupted by additive noise of different strengths is illustrated in Fig. 1. It is clearly shown that the same algorithm using $\tilde{r}$, defined in (11), converges much slower than the one using $\tilde{r}^*$, defined (10), on different dimensions.

C. Comparison with state-of-the-art derivative-free optimization algorithms

The Python3 library *nevergrad* [8] is an open source software for derivative-free optimization, which includes a set of benchmark functions, various derivative-free optimization algorithms and some noise models [8]. We compare (Fig. 2) our approximate resampling rule (10) to the implementations of two state-of-the-art algorithms, simultaneous perturbation

TABLE III

OPTIMIZED PARAMETERS (COLUMNS 3-8) AND APPROXIMATE SLOPES (COLUMN 9, AS IN (9): THE LOWER, THE FASTER) IN DIFFERENT TEST CASES WITH OPTIMIZED PARAMETERS USING 600 ITERATIONS DURING META-OPTIMIZATION (UPPER LEVEL) AND $5e5$ EVALUATIONS DURING THE TUNING (LOWER LEVEL). IN THIS TABLE, THE TERM "SLOPE" REFERS TO THE EVALUATION OF OBJECTIVE FUNCTIONS DEFINED IN TABLE II. THE "OPTIMIZED SLOPE" (COLUMN 9) REFERS TO THE APPROXIMATE SLOPES OBTAINED AFTER EACH META-OPTIMIZATION PHASE. THE BEST APPROXIMATE SLOPE IN EACH TEST CASE IS PRESENTED IN BOLDFACE. AFTER THE OPTIMIZATION PHASE, WE APPLY THE DETERMINED RESAMPLING RULES USING THE OPTIMIZED PARAMETERS (COLUMNS 3-8) TO THE 7 TEST CASES (NO MORE TUNING) AND COLUMNS 10-16 PRESENT THE OBTAINED AVERAGED SLOPES OVER 11 TRIALS. GRAY COLUMNS ARE SLOPES OBTAINED BY APPLYING THE PARAMETERS OPTIMIZED ON TEST CASE i TO THEMSELVES, $\forall i \in \{1, \ldots, 7\}$. A SLOPE OF 0.0000 MEANS THAT THE OPTIMIZATION TRIAL FAILS, THUS, DIVERGENCES.

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16
		Optimization phase							**Test phase**						
		Optimized parameters						Optimized slope	Slopes obtained in different test cases with parameters obtained after *optimization phase* (columns 3-8)						
Test case	Resampling rule	Non-adaptive part		step-size part	Weight	Bernstein module									
		ζ	κ	η	ϱ	α	β		$\mathcal{T}_1$	$\mathcal{T}_2$	$\mathcal{T}_3$	$\mathcal{T}_4$	$\mathcal{T}_5$	$\mathcal{T}_6$	$\mathcal{T}_7$
$\mathcal{T}_1$	r_{NA}	1.1717	1.0440	-	0.4570	-	-	**-1.1442**	-1.1342	-0.3192	-0.1071	-0.7250	-0.6178	-0.2050	-0.5141 *
	r_{σ}	0.8443	2.3609	0.6751	-	-	-	-1.1370	-1.1390	-0.2306	-0.0500	-0.6801	-0.5930	-0.1433	-0.4747
	$r_{NA,\sigma}$	0.6473	3.6374	1.1945	0.2813	-	-	-1.1356	-1.1424	-0.1720	-0.0316	-0.6557	-0.5865	-0.1028	-0.4476
	$r_{NA,\tau_{EBB}}$	1.5291	1.1840	-	0.0835	6.9498	6.1939	-1.1096	-0.8451	-0.2850	-0.0820	-0.5797	-0.4624	-0.1829	-0.4079
	$r_{\sigma,\tau_{EBB}}$	0.7205	0.3754	0.0170	-	11.6889	8.0762	-1.0561	-1.0725	-0.2374	-0.0555	-0.6526	-0.5586	-0.1495	-0.4497
	$r_{NA,\sigma,\tau_{EBB}}$	1.1218	1.0168	0.2304	0.7791	1.4680	2.5756	-1.1186	-0.9607	-0.2615	-0.0666	-0.6096	-0.5125	-0.1647	-0.4297
$\mathcal{T}_2$	r_{NA}	1.6595	0.9898	-	0.2338	-	-	-0.3099	-0.6180	-0.3350	-0.1070	-0.4679	-0.3628	-0.2232	-0.3499
	r_{σ}	Inf	0.0000	0.0000	-	-	-	0.0000	-	-	-	-	-	-	-
	$r_{NA,\sigma}$	1.1809	0.9443	1.8227	0.7220	-	-	**-0.3392**	-0.5435	-0.3328	-0.1135	-0.4503	-0.3353	-0.2213	-0.3308
	$r_{NA,\tau_{EBB}}$	1.2662	1.0015	-	0.9240	1.7021	1.1816	-0.3110	-0.5844	-0.2943	-0.0728	-0.4426	-0.3372	-0.1875	-0.3253
	$r_{\sigma,\tau_{EBB}}$	2.2133	1.3473	0.6613	-	1.4860	1.2819	-0.3151	-0.4471	-0.3064	-0.0935	-0.3818	-0.2763	-0.2086	-0.2921
	$r_{NA,\sigma,\tau_{EBB}}$	1.8551	1.0107	0.0957	0.1316	2.8414	1.1930	-0.3231	-0.7184	-0.3017	-0.0830	-0.5179	-0.4021	-0.1917	-0.3665
$\mathcal{T}_3$	r_{NA}	1.4061	1.0195	-	0.7813	-	-	-0.1118	-0.6037	-0.3326	-0.1080	-0.4730	-0.3580	-0.2151	-0.3458
	r_{σ}	2.9144	1.3364	0.3463	-	-	-	-0.0875	-0.4625	-0.3209	-0.1086	-0.3993	-0.2834	-0.2089	-0.3003
	$r_{NA,\sigma}$	0.0694	15.4412	6.4953	0.3066	-	-	**-0.1170**	-0.2049	-0.1922	-0.1133	-0.2005	-0.1609	-0.1592	-0.1718
	$r_{NA,\tau_{EBB}}$	2.1528	0.8545	-	0.0500	22.2492	4.9408	-0.1016	-0.2669	-0.2324	-0.0926	-0.2513	-0.1843	-0.1692	-0.2009
	$r_{\sigma,\tau_{EBB}}$	2.2373	1.3269	1.0109	-	1.0125	1.1082	-0.0905	-0.3211	-0.2725	-0.0945	-0.2953	-0.2019	-0.1806	-0.2265
	$r_{NA,\sigma,\tau_{EBB}}$	1.2713	1.0338	16.8935	0.9008	1.4168	6.9599	-0.0937	-0.0192	-0.0202	-0.0133	-0.0181	-0.0180	-0.0153	-0.0136
$\mathcal{T}_4$	r_{NA}	1.1323	0.9990	-	0.6638	-	-	**-0.7309**	-1.1426	-0.3226	-0.0941	-0.7378	-0.6206	-0.2082	-0.5189 *
	r_{σ}	1.6566	1.4716	0.2148	-	-	-	-0.7225	-1.1269	-0.2879	-0.0815	-0.7115	-0.6085	-0.1876	-0.5017
	$r_{NA,\sigma}$	1.2354	1.5549	0.4672	0.2325	-	-	-0.6927	-1.1293	-0.2804	-0.0816	-0.7032	-0.6124	-0.1804	-0.4985
	$r_{NA,\tau_{EBB}}$	1.3750	1.0980	-	0.1472	2.5663	1.0607	-0.6869	-1.0650	-0.2702	-0.0744	-0.6630	-0.5649	-0.1691	-0.4641
	$r_{\sigma,\tau_{EBB}}$	1.8114	1.5004	0.2068	-	1.6737	4.2571	-0.6911	-1.1075	-0.2663	-0.0773	-0.6856	-0.5924	-0.1742	-0.4830
	$r_{NA,\sigma,\tau_{EBB}}$	1.1005	1.4947	14.0566	0.7220	7.0467	3.4607	-0.6483	-0.0390	-0.0316	-0.0226	-0.0263	-0.0308	-0.0202	-0.0226
$\mathcal{T}_5$	r_{NA}	1.1978	1.0754	-	0.4097	-	-	**-0.6162**	-1.1402	-0.3113	-0.0942	-0.7246	-0.6127	-0.2019	-0.5145
	r_{σ}	Inf	0.0000	0.0000	-	-	-	0.0000	-	-	-	-	-	-	-
	$r_{NA,\sigma}$	0.5736	4.2415	1.2609	0.2614	-	-	-0.5842	-1.1420	-0.1563	-0.0351	-0.6458	-0.5811	-0.0921	-0.4411
	$r_{NA,\tau_{EBB}}$	1.1757	1.0350	-	0.4860	10.2056	9.2748	-0.6008	-0.4637	-0.3018	-0.0748	-0.3881	-0.2699	-0.1881	-0.2794
	$r_{\sigma,\tau_{EBB}}$	0.5563	0.4438	0.1798	-	2.9083	1.0856	-0.5565	-1.0676	-0.2206	-0.0398	-0.6489	-0.5533	-0.1368	-0.4411
	$r_{NA,\sigma,\tau_{EBB}}$	1.1353	1.5172	1.0748	0.6066	2.7049	1.1533	-0.5963	-0.4673	-0.3028	-0.0780	-0.3885	-0.2678	-0.1882	-0.2829
$\mathcal{T}_6$	r_{NA}	1.2037	0.9199	-	0.9581	-	-	-0.2169	-0.7601	-0.3271	-0.1037	-0.5479	-0.4340	-0.2171	-0.3974
	r_{σ}	2.5262	1.3205	0.4072	-	-	-	-0.2193	-0.5023	-0.3290	-0.1110	-0.4147	-0.2985	-0.2186	-0.3125
	$r_{NA,\sigma}$	1.6885	0.9915	0.0276	0.1966	-	-	**-0.2412**	-0.6234	-0.3284	-0.1097	-0.4738	-0.3523	-0.2154	-0.3507
	$r_{NA,\tau_{EBB}}$	1.3726	0.9149	-	0.4619	14.7646	1.1179	-0.2014	-0.2343	-0.2215	-0.0933	-0.2272	-0.1602	-0.1542	-0.1809
	$r_{\sigma,\tau_{EBB}}$	Inf	0.0000	0.0000	-	1.0000	Inf	0.0000	-	-	-	-	-	-	-
	$r_{NA,\sigma,\tau_{EBB}}$	2.4229	1.2582	0.4247	0.0260	25.3590	7.4425	-0.1952	-0.4610	-0.3160	-0.0885	-0.3870	-0.2751	-0.2044	-0.2915
$\mathcal{T}_7$	r_{NA}	1.7980	1.2524	-	0.0355	-	-	**-0.5166**	-1.1185	-0.3076	-0.0914	-0.7127	-0.6076	-0.2076	-0.5036
	r_{σ}	2.0474	1.3376	0.0124	-	-	-	-0.5098	-1.1270	-0.2950	-0.0893	-0.7129	-0.6073	-0.1923	-0.5033
	$r_{NA,\sigma}$	1.1084	1.1161	2.5190	0.8482	-	-	-0.5126	-1.1286	-0.3087	-0.0975	-0.7221	-0.6137	-0.2069	-0.5119
	$r_{NA,\tau_{EBB}}$	1.8575	1.2620	-	0.0269	-	-	-0.4848	-0.0148	-0.0079	-0.0100	-0.0125	-0.0105	-0.0079	-0.0117
	$r_{\sigma,\tau_{EBB}}$	1.8479	1.4783	0.1886	-	1.0766	26.3923	-0.4763	-1.1055	-0.2744	-0.0782	-0.6885	-0.5937	-0.1713	-0.4878
	$r_{NA,\sigma,\tau_{EBB}}$	1.1857	1.2070	0.3758	0.4669	1.0890	1.4013	-0.4863	-0.9006	-0.2689	-0.0632	-0.5808	-0.4789	-0.1662	-0.4140

stochastic approximation (SPSA) and TBPSA[1], a simplified version of the population size controlled, comparison-based, self-adaptive ES (termed pcCMSA-ES) [26] which reached $slope(SR) = -1$ on the same noisy function studied by [19] - with an algorithm not satisfying the simplicity assumption. The symmetric noise model is $f_{noisy}(x) = f(x) + S \times (f(x + \mathcal{N}_d) - f(x)) \times \mathcal{N}_1$, and the asymmetric one is $f_{disym}(x) = f(x) + S \times (1 + f(x)) \times \chi_{x_0>0} \times (f(x + \mathcal{N}_d) - f(x)) \times \mathcal{N}_1$, where $\mathcal{N}_d$ is a standard Gaussian random variable in dimension d and we test noise levels $S \in \{1, 10, 100\}$. Thus, both a prior and posterior noise sources are considered. Other resampling formulas used in the comparison are ablations of our formula.

The difference between (10) and other formulas (detailed in the caption of Fig. 2) are moderate, the key message from Fig. 2 is more the comparison with the population control method TBPSA and SPSA and the fact that overall we outperform the *O101* rule from [7].

VI. CONCLUSION

In this work, we consider resampling rules used in evolutionary algorithms for solving noisy optimization problems. We combine some non-adaptive and adaptive resampling rules for noisy evolutionary optimization. In terms of adaptive rules, a stopping rule by Empirical Bernstein Bound is also tested.

[1]More details of the algorithms and their implementations can be found at https://github.com/facebookresearch/nevergrad.

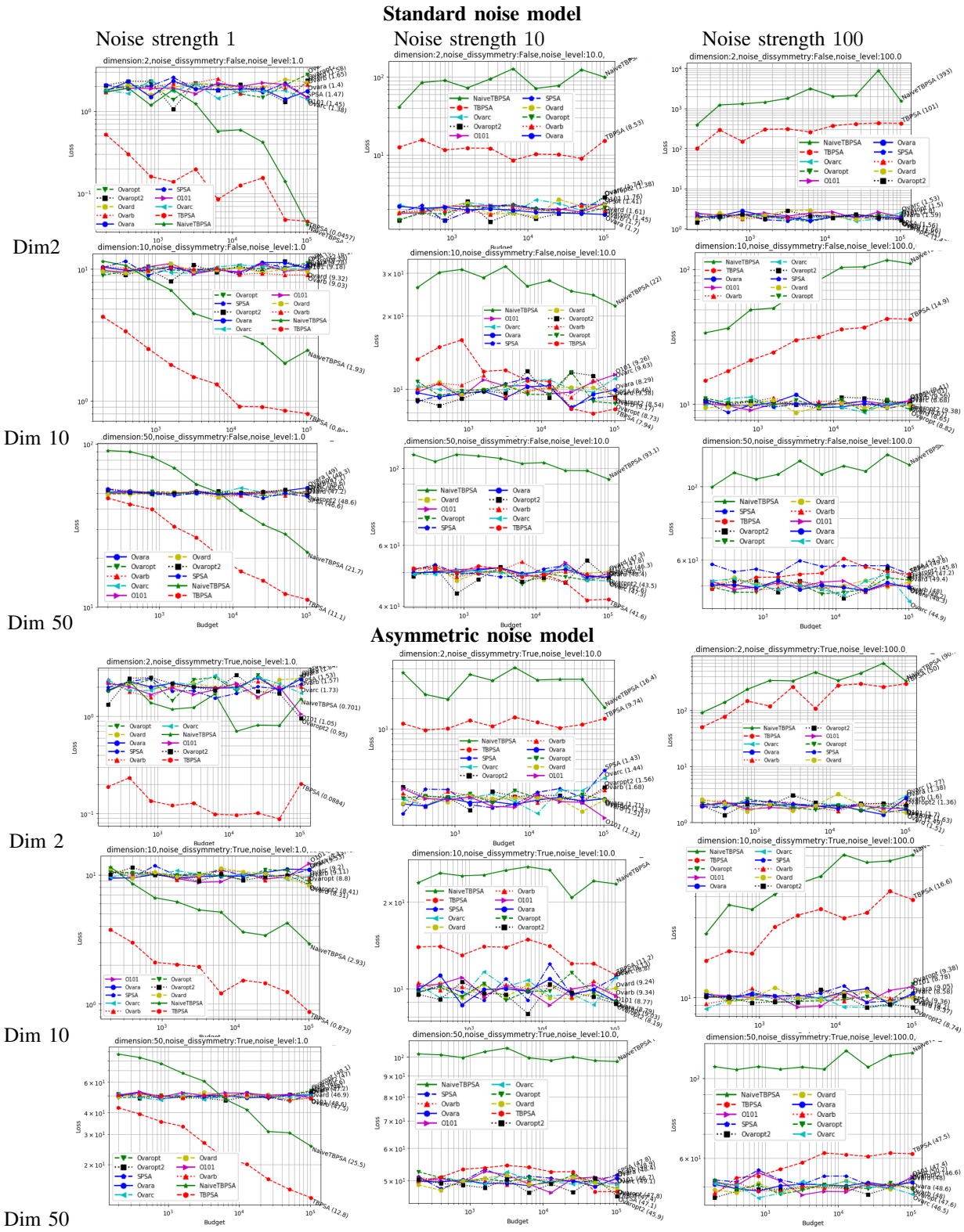

Fig. 2. Comparison of our proposed rule (10) (labelled as *ovaropt2*), ablations of our rule (labelled as *ovara*= $\lceil\sqrt{n}\rceil$, *ovarb*= n, *ovarc*= $\lceil\sqrt{\sqrt{n}}\rceil$, *ovard*= $\lceil\sqrt{n \cdot d}\rceil$ and (11): *ovaropt*= $\lceil\sqrt{\frac{n}{d}}\rceil$), the *O101*= $\lceil n^{1}.01\rceil$ rule in [7] and other algorithms (labelled as *NaiveTBPSA*, *TBPSA* and *SPSA*) in *nevergrad*. We see that population control perform better than resampling rules (including ours) in some cases, but its performance degrades with high noise (rightmost column) or misleading noise model (bottom block).

TABLE IV

AVERAGED APPROXIMATE SLOPES (THE LOWER, THE BETTER) IN DIFFERENT TEST CASES WITH THE APPROXIMATE RESAMPLING RULE (10). THE LAST 3 COLUMNS ILLUSTRATE THE APPROXIMATE SLOPES, DEFINED IN (8), OBTAINED AFTER CERTAIN NUMBERS OF EVALUATIONS FOR THE NOISY FUNCTIONS f_1, f_2 AND f_3 OF VARIOUS DIMENSIONS, WITH THEIR NOISE STRENGTH INDICATED IN THE FIRST COLUMN, RESPECTIVELY.

Noise	d	Approximate slope		
		After $T = 5e5$	After $T = 1e7$	After $T = 1e9$
$1e^{-6}$	2	-1.4538 $\pm$ 0.0662	-1.3354 $\pm$ 0.1391	-1.1406 $\pm$ 0.0634
	4	-1.3570 $\pm$ 0.0724	-1.2136 $\pm$ 0.0402	-1.0619 $\pm$ 0.0352
	8	-1.2895 $\pm$ 0.0356	-1.1581 $\pm$ 0.0414	-1.0072 $\pm$ 0.0294
	16	-1.1906 $\pm$ 0.0291	-1.0792 $\pm$ 0.0237	-0.9483 $\pm$ 0.0195
	32	-1.1034 $\pm$ 0.0426	-1.0215 $\pm$ 0.0181	-0.8955 $\pm$ 0.0106
	64	-0.9973 $\pm$ 0.0213	-0.9474 $\pm$ 0.0087	-0.8499 $\pm$ 0.0067
0.05	2	-0.6434 $\pm$ 0.0911	-0.6164 $\pm$ 0.0623	-0.6206 $\pm$ 0.0919
	4	-0.5677 $\pm$ 0.0551	-0.5269 $\pm$ 0.0431	-0.5433 $\pm$ 0.0457
	8	-0.4641 $\pm$ 0.0461	-0.4750 $\pm$ 0.0267	-0.4817 $\pm$ 0.0278
	16	-0.3769 $\pm$ 0.0301	-0.4170 $\pm$ 0.0177	-0.4329 $\pm$ 0.0206
	32	-0.3006 $\pm$ 0.0110	-0.3419 $\pm$ 0.0177	-0.3748 $\pm$ 0.0106
	64	-0.2251 $\pm$ 0.0140	-0.2750 $\pm$ 0.0093	-0.3299 $\pm$ 0.0104
1	2	-0.4142 $\pm$ 0.0668	-0.4558 $\pm$ 0.0865	-0.4443 $\pm$ 0.0510
	4	-0.3220 $\pm$ 0.0655	-0.3569 $\pm$ 0.0390	-0.3936 $\pm$ 0.0256
	8	-0.2531 $\pm$ 0.0365	-0.2956 $\pm$ 0.0359	-0.3417 $\pm$ 0.0225
	16	-0.1492 $\pm$ 0.0236	-0.2217 $\pm$ 0.0296	-0.2950 $\pm$ 0.0206
	32	-0.0942 $\pm$ 0.0183	-0.1596 $\pm$ 0.0152	-0.2353 $\pm$ 0.0120
	64	-0.0048 $\pm$ 0.0157	-0.0919 $\pm$ 0.0083	-0.1828 $\pm$ 0.0057

The design and tuning of resampling formulas is realised by automatic parameter tuning of the parameters in the combined formula and the stopping rule, equipped with a rounding of obtained coefficients. When tuning the rules, their convergence rates on strongly noisy optimization problems, in which the variance of the noise does not vanish in the neighborhood of the optimum, are used as the performance indicator.

We proposed a parameter-free, simple but effective resampling rule $\tilde{r}^* = \lceil 1.1^{\frac{n}{d}} \cdot \max\{1, \sqrt{\frac{n}{d}}\} \rceil$ at iteration n (d is the problem dimension). Our new resampling rule performs well compared to various ablations, but the difference is minor (Fig. 2). We note the importance of the exponential term $1.1^{\frac{n}{d}}$ (Section V-B - this is consistent with [7]); the lack of benefit associated to the Bernstein stopping rule (Section III-E); and the lack of dependencies in the step-size (Section III-D). We also note that the population control method of *nevergrad* outperforms our resampling rules for simple noise models with moderate strength, but our algorithm tends to outperform it in the following three cases: when the noise is strong; when the noise model is complicated; or when the dimension is low.

ACKNOWLEDGMENT

The authors would like to thank Prof. Xin Yao for his valuable comments.

REFERENCES

[1] S. Astete-Morales, J. Liu, and O. Teytaud, "Log-log convergence for noisy optimization," in *International Conference on Artificial Evolution (Evolution Artificielle)*. Springer, 2013, pp. 16–28.

[2] Y. Jin and J. Branke, "Evolutionary optimization in uncertain environments-a survey," *IEEE Transactions on evolutionary computation*, vol. 9, no. 3, pp. 303–317, 2005.

[3] P. Rakshit, A. Konar, and S. Das, "Noisy evolutionary optimization algorithms–a comprehensive survey," *Swarm and Evolutionary Computation*, vol. 33, pp. 18–45, 2017.

[4] M.-L. Cauwet, J. Liu, B. Rozière, and O. Teytaud, "Algorithm portfolios for noisy optimization," *Annals of Mathematics and Artificial Intelligence*, vol. 76, no. 1-2, pp. 143–172, 2016.

[5] S. Astete-Morales, M.-L. Cauwet, J. Liu, and O. Teytaud, "Simple and cumulative regret for continuous noisy optimization," *Theoretical Computer Science*, vol. 617, pp. 12–27, 2016.

[6] J. Liu, D. L. Saint-Pierre, and O. Teytaud, "A mathematically derived number of resamplings for noisy optimization," in *Companion-Genetic and Evolutionary Computation Conference*. ACM, 2014, pp. 61–62.

[7] S.-Y. Chiu, C.-N. Lin, J. Liu, T.-C. Su, F. Teytaud, O. Teytaud, and S.-J. Yen, "Differential evolution for strongly noisy optimization: Use 1.01^n resamplings at iteration n and reach the $-\frac{1}{2}$ slope," in *2015 IEEE Congress on Evolutionary Computation*. IEEE, 2015, pp. 338–345.

[8] J. Rapin and O. Teytaud, "Nevergrad - A gradient-free optimization platform," https://GitHub.com/FacebookResearch/Nevergrad, 2018.

[9] C. Qian, Y. Yu, and Z.-H. Zhou, "Analyzing evolutionary optimization in noisy environments," *Evolutionary computation*, vol. 26, no. 1, pp. 1–41, 2018.

[10] C. Qian, Y. Yu, K. Tang, Y. Jin, X. Yao, and Z.-H. Zhou, "On the effectiveness of sampling for evolutionary optimization in noisy environments," *Evolutionary computation*, vol. 26, no. 2, pp. 237–267, 2018.

[11] D. V. Arnold and H.-G. Beyer, "Efficiency and mutation strength adaptation of the (μ/μ_I, λ)-ES in a noisy environment," in *Parallel Problem Solving from Nature PPSN VI*. Springer, 2000, pp. 39–48.

[12] H.-G. Beyer, "Evolutionary algorithms in noisy environments: Theoretical issues and guidelines for practice," *Computer methods in applied mechanics and engineering*, vol. 186, no. 2-4, pp. 239–267, 2000.

[13] D. V. Arnold and H.-G. Beyer, "Investigation of the (μ,λ)-ES in the presence of noise," in *Evolutionary Computation, 2001. Proceedings of the 2001 Congress on*, vol. 1. IEEE, 2001, pp. 332–339.

[14] ——, "Local performance of the (1+1)-ES in a noisy environment," *IEEE Transactions on Evolutionary Computation*, vol. 6, no. 1, pp. 30–41, 2002.

[15] M. Jebalia and A. Auger, "On multiplicative noise models for stochastic search," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2008, pp. 52–61.

[16] M.-L. Cauwet, J. Liu, and O. Teytaud, "Algorithm portfolios for noisy optimization: Compare solvers early," in *International Conference on Learning and Intelligent Optimization*. Springer, Cham, 2014, pp. 1–15.

[17] D. V. Arnold and H.-G. Beyer, "A general noise model and its effects on evolution strategy performance," *IEEE Transactions on Evolutionary Computation*, vol. 10, no. 4, pp. 380–391, 2006.

[18] H.-G. Beyer, "Mutate large, but inherit small! On the analysis of rescaled mutations in (1,$lambda$)-ES with noisy fitness data," in *Parallel Problem Solving from Nature, PPSN V*. Springer, 1998, pp. 109–118.

[19] S. Astete-Morales, M.-L. Cauwet, and O. Teytaud, "Evolution strategies with additive noise: A convergence rate lower bound," in *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. ACM, 2015, pp. 76–84.

[20] T. Friedrich, T. Kötzing, M. S. Krejca, and A. M. Sutton, "The compact genetic algorithm is efficient under extreme gaussian noise," *IEEE Transactions on Evolutionary Computation*, vol. 21, no. 3, pp. 477–490, 2017.

[21] P. Rolet and O. Teytaud, "Bandit-based estimation of distribution algorithms for noisy optimization: Rigorous runtime analysis," in *International Conference on Learning and Intelligent Optimization*. Springer, 2010, pp. 97–110.

[22] V. Heidrich-Meisner and C. Igel, "Hoeffding and bernstein races for selecting policies in evolutionary direct policy search," in *ICML '09: Proceedings of the 26th Annual International Conference on Machine Learning*. New York, NY, USA: ACM, 2009, pp. 401–408.

[23] N. Hansen, S. Niederberger, L. Guzzella, and P. Koumoutsakos, "A method for handling uncertainty in evolutionary optimization with an application to feedback control of combustion," *IEEE Transactions on Evolutionary Computation*, vol. 13, no. 1, pp. 180–197, 2009.

[24] J.-Y. Audibert, R. Munos, and C. Szepesvári, "Tuning bandit algorithms in stochastic environments," in *Algorithmic Learning Theory*. Springer, 2007, pp. 150–165.

[25] V. Mnih, C. Szepesvári, and J.-Y. Audibert, "Empirical bernstein stopping," in *Proceedings of the 25th international conference on Machine learning*. ACM, 2008, pp. 672–679.

[26] M. Hellwig and H.-G. Beyer, "Evolution under strong noise: A self-adaptive evolution strategy can reach the lower performance bound-the pccmsa-es," in *International Conference on Parallel Problem Solving from Nature*. Springer, 2016, pp. 26–36.