Differential Evolution for Strongly Noisy Optimization: Use 1.01^n Resamplings at Iteration n and Reach the -1/2 Slope

Shih-Yuan Chiu, Ching-Nung Lin, Jialin Liu, Tsang-Cheng Su, Fabien Teytaud, Olivier Teytaud, Shi-Jim Yen

Differential Evolution for Strongly Noisy Optimization: Use 1.01^n Resamplings at Iteration n and Reach the $-\frac{1}{2}$ Slope

Shih-Yuan Chiu*, Ching-Nung Lin*, Jialin Liu†, Tsan-Cheng Su*, Fabien Teytaud‡, Olivier Teytaud†
Shi-Jim Yen*,
* CSIE, in National Dong-Hwa University, Hualien, Taiwan
† TAO, Inria, Univ. Paris-Sud, UMR CNRS 8623, France
‡ Univ. Lille Nord de France, France

***Abstract*—This paper is devoted to noisy optimization in case of a noise with standard deviation as large as variations of the fitness values, specifically when the variance does not decrease to zero around the optimum. We focus on comparing methods for choosing the number of resamplings. Experiments are performed on the differential evolution algorithm. By mathematical analysis, we design a new rule for choosing the number of resamplings for noisy optimization, as a function of the dimension, and validate its efficiency compared to existing heuristics.**

***Keywords*—Noisy Optimization, Differential Evolution, Resampling**

I. Introduction

Differential Evolution (DE)[1] is a well-known algorithm with an ability to handle second order information without expensive Hessian computations. Essentially, it is usually slower than CMA-ES or CMSA in terms of number of function evaluations for randomly rotated ill conditioned functions[2] but "vastly outperforms" CMA for less rotated functions[3]. It is a crucial component of [4] which won the LSCO2013 competition. Another less widely cited advantage (invisible for comparisons for a fixed number of evaluations) which is critical in high dimension is the small internal computation time. It also won many competitions[5].

Research in noisy optimization is very important, because (i) it is relevant in real applications and (ii) there are not so many conclusive results in some important cases. In particular, there are few papers focusing on strongly noisy optimization, in which the standard deviation of noise has the same order of magnitude as differences between fitness values. We point out that this is not equivalent to additive noise: it is a form of additive noise, but we have an additional requirement that the order of magnitude of fitness values is the same as the standard deviation of noise. This avoids the case of noisy objective functions with very little noise, which might be of some importance as well but which is not the focus of the present paper.

Noisy optimization occurs frequently in power systems, games[6] and Direct Policy Search[7]. For example, in games, a success rate of 60% of program 1 against program 2 leads to a standard deviation close to $\frac{1}{2}$: this is strong noise. As long as Direct Policy Search deals with a success rate, the situation is similar to games: when the expected reward (which is the success rate) is $p \in [0, 1]$, and if the objective function is the outcome of a game, then the objective function is a Bernoulli random variable[1] with parameter p, and the standard deviation is $\sqrt{p(1-p)}$. Then, as long as p and $1-p$ are far from zero, $\sqrt{p(1-p)}$ has the same order of magnitude as fitness values. The case of standard deviations converging to zero close to the optimum is then the case of a zero risk close to the optimum, which is a limited case.

Section II presents the state of the art, Section III presents differential evolution and our proposed counterparts for noisy optimization. Section IV presents experimental results.

II. State of The Art: Noisy Optimization

In this paper, the fitness function is corrupted by noise. Therefore, one single value of $f(x, w)$ might not be a good estimate of $\mathbb{E}f(x, w)$. It is known that evolution strategies might not converge in the presence of noise[8], [9], at least without special treatment. As long as the noise is small in comparison to the differences between fitness values, everything goes smoothly; but eventually, the algorithm stops converging. Resampling fitness values in order to reduce the variance is a natural solution[10]: instead of evaluating fitness values once, evaluate them r_n times (each), at iteration n, in order to reduce the noise. Yet there is no clearly established rule for choosing the number of resamplings.

More formally, with $^{(i)}$ denoting the i^{th} independent copy of a random variable, the N^{th} resampling of $f(x, w)$ denotes $\frac{1}{N}\sum_{i=1}^{N} f(x, w^{(i)})$. Differential evolution involves only pairwise comparisons. The comparisons are performed on the empirical evaluation, so the candidate p' outperforms p after N resamplings if $\frac{1}{N}\sum_{i=1}^{N} f(p, w^{(i)}) < \frac{1}{N}\sum_{i=1}^{N} f(p', (w)^{(i+N)})$.

A. Non-adaptive rules

[10] has proved mathematically that a non-adaptive rule with exponential number of resamplings can lead to *log-log* convergence, i.e. the logarithm of the distance to the optimum typically scales linearly with the logarithm of number of evaluations. [10] has also shown experimentally that a non-adaptive rule with polynomial number of resamplings can lead to the *log-log* convergence.

[1] A Bernoulli random variable has outcome 1 with probability p and 0 with probability $1-p$.

B. Adaptive rules

Bernstein races[11] are aimed at comparing probability distributions, thanks to empirical samples. The principle consists in sampling each distribution until a statistically significant difference between expectations is detected. It makes sense to use Bernstein races for comparing a population of search points; we can compare individuals until we have a statistically significant ranking. Bernstein races have been applied for noisy optimization in [12]. A problem arises when expected fitness values are equal; in such a case, the algorithm might not converge, just because the number of resamplings becomes infinite. This is because, with probability at least $1-\delta$ for some $\delta < 1$, the Bernstein race runs until it finds a significant difference, which is eternity when there is no significant difference.

III. Differential evolution & noisy differential evolution

A. Differential evolution

The Differential Evolution (DE) algorithm[1] is an optimization algorithm which operates in continuous search spaces. DE belongs to the family of Evolutionary Algorithms (EAs). It does not need the fitness function to be differentiable. It is based on the four main steps of EAs : initialization, mutation, recombination and selection. This process is iterated until an acceptable solution is found or until a certain time limit is reached. We use Differential Evolution as described in Alg. 1. D is the dimension of the search space. Many variants

Algorithm 1 DE/rand/2: Pseudo-code of Differential Evolution. For $j \in \{1,\dots,D\}$, $(x)_j$ denotes the j^{th} coordinate of a vector $x \in \mathbb{R}^D$.

Input $F \in [0,2]$: Differential weight
Input $Cr \in [0,1]$: Crossover probability
Input λ: Population size
Initialize $p_1,\dots,p_\lambda$ uniformly in the bounded search space
while not finished **do**
  for $i \in \{1,\dots,\lambda\}$ **do**
    Randomly draw a, b, c, d and e distinct in $\{1,\dots,i-1,i+1,\dots,\lambda\}$
    Define

$$p'_i = p_a + F(p_b - p_c) + F(p_d - p_e) \qquad (1)$$

    Randomly draw $R \in \{1,\dots,D\}$
    for $j \in \{1,\dots,D\}$, **do**
      if $rand < Cr$ or $j == R$ **then**
        $(p''_i)_j = (p'_i)_j$
      else
        $(p''_i)_j = (p_i)_j$
      end if
    end for
    $p_i = best(p_i, p''_i)$ (keep p_i in case of tie)
  end for
end while

exist, including self-adaptive parameters[13], [14], [15], [16] and meta-heuristics for choosing parameters[17]. The mutation step is a crucial point and several types exist. Most well known are:

- DE/best/1 [18] : $p'_i = p_{best} + F(p_b - p_c)$;
- DE/best/2 [18] : $p'_i = p_{best} + F(p_b - p_c) + F(p_d - p_e)$;
- DE/rand/1 [18] : $p'_i = p_a + F(p_b - p_c)$;
- DE/rand/2 (Eq. 1);

where p_{best} is the best point in the current population, p_a, p_b, p_c, p_d and p_e are distinct points randomly chosen in the current population. In this paper we focus on "DE/rand/2". We use $\lambda = 100$ particles, $F = 0.7$ and $Cr = 0.5$.

B. Noisy differential evolution: state of the art

DE is simple, handles ill conditioning correctly, and spends very little time in internal computation. However, the performance of DE is unstable when the fitness function is corrupted by noise[19].

Resamplings and threshold can be used to deal with noise. [20] studied an improved DE (DE/rand/1) algorithm where the scalar factor used to weigh the difference vector has been randomized, called Differential Evolution with Random Scale Factor (DE-RSF). A threshold based Selection Strategy, aimed at overcoming the noise through resampling, has been combined into the DE-RSF, namely DE-RSF-TS. Another variant is DE-RSF with Stochastic Selection, namely DE-RSF-SS, with which the offspring is selected as the parent of next generation with probability calculated by the ratio of the fitness of parent to the one of the offspring. [20] showed that both DE-RSF-TS and DE-RSF-SS performed worse than DE for noise-free functions. But in noisy cases, these two improved algorithms (i) can more efficiently find the global optimum and (ii) are more efficient than the DE/Rand/1/Exp, the canonical PSO (Particle Swarm Optimization) and the standard real coded EA on classical benchmarks corrupted by zero-mean Gaussian noise.

[21], [22] proposed a new Opposition-Based DE (ODE) algorithm, which uses Opposition-Based Optimization for population initialization, generation jumping and improving population's best individual. ODE has a concordant performance for both noise-free case and noisy case with constant variance of noise.

[23] combined the Optimal Computing Budget Allocation (OCBA) technique and the Simulated Annealing (SA) algorithm into differential evolution. Their hybrid algorithm is designed for noisy and uncertain environments inspired from real world applications. [23] concludes that, by incorporating both SA and OCBA into DE, the performance is improved for fitness functions corrupted by large noise.

We refer to [24] for more on noisy optimization and differential evolution.

C. Non-adaptive and adaptive noisy differential evolution

1) Non-adaptive numbers of resamplings: We tested various rules for non-adaptive numbers of resamplings: $N_{linear} = n$; $N_{square\ root} = \sqrt{n}$; $N_{2exp} = 2^n$; $N_{constant} = 1$; $N_{scale} = \lceil D^{-2} \exp(\frac{4n}{5D}) \rceil$; where n is the generation index.After preliminary testing, we removed some of them for the clarity of graphs. The N_{scale} formula was derived in [25] based on scale analysis.

2) Adaptive numbers of resamplings: We propose here methods inspired from [12], [26], adapted to differential evolution. It might make sense to resample until some statistical test becomes positive. For example, we might do a test based on standard deviation after each batch of 1000 resamplings. Such a batch size makes sense for parallelization, and for reducing the cumulative risk of false positive. We therefore define, for a pair of search points x, x' to be compared:

$$\begin{aligned} y_m &= \sum_{i=1}^{1000m} f(x, w^{(i)}), \\ y'_m &= \sum_{i=1}^{1000m} f(x', (w')^{(i)}) \\ \delta_m &= y_m - y'_m, \quad \mu_m = \frac{1}{m}\sum_{i=1}^{m} \delta_i, \\ \sigma_m^2 &= \frac{1}{m}\sum_{i=1}^{m} (\delta_i - \mu_m)^2 \\ N_{adaptive} &= \min_{m \in \{2,3,4,\dots\}} \{1000m; |\mu_m| > \frac{\sigma_m}{\sqrt{m-1}}\} \quad (2) \\ N_{\substack{enhanced \\ adaptive}} &= \min\{ \min_{m \in \{2,3,4,\dots\}} \{10^3 m; |\mu_m| > \frac{\sigma_m}{\sqrt{m-1}}\}, \\ &\quad \lceil \frac{2^n}{10^3} \rceil \times 10^3\} \quad (3) \end{aligned}$$

This means that we evaluate points until there is a statistically significant difference. However, this is not a rigorous test, because we test repeatedly, after each group of 1000 resamplings. Even if each test is guaranteed (e.g. with confidence 95%), the whole set of tests is not guaranteed with confidence 95% but with confidence $\max(0, 100-5P)\%$ after P tests. In particular, even if x and x' are equal, the procedure will eventually stop (more details below on this).

It is for sure possible to do tests differently. One can repeat the test and decrease the risk threshold so that the overall probability of misranking is always less than a fixed δ. But in such a case, we possibly get infinite loops when we meet a plateau. We keep our version above, and consider it as an adaptation ensuring almost sure halting. Indeed, [12] also includes a rule for ensuring the finiteness of the number of resamplings.

Let us formalize below the almost sure halting property.

Proposition 1 (Almost sure halting of this procedure) *If $f(x, w)$ and $f(x', w)$ have finite positive variance, then $N_{adaptive}$ is almost surely finite.*

Proof: If $\mathbb{E}f(x, w) \neq \mathbb{E}f(x', w)$, the result is immediate by the law of large numbers, applicable due to finite variance.

If $\mathbb{E}f(x, w) = \mathbb{E}f(x', w)$, then by the law of the iterated logarithm, the supremum over $i \in \{2, \dots, N\}$ of $\frac{\sqrt{i-1}\mu_i}{\sigma_i}$ is almost surely going to infinity as $N \to \infty$ - therefore, at some point Eq. 2 holds, so almost surely N is finite. ■

Let us formalize the concept of resampling rule. A resampling rule evaluates several times the fitness of two search points x and x'. $\hat{\mathbb{E}}f(x, w)$ is the average fitness value for x, and $\hat{\mathbb{E}}f(x', w)$ is the average fitness value for x'. At some point, the rule decides to stop reevaluating. It then outputs a result, which is either "x is better than x'", or "x' is better than x", or "x and x' have the same expected fitness", depending on the sign of $\hat{\mathbb{E}}f(x, w) - \hat{\mathbb{E}}f(x', w)$. We point out the following counterpart of Proposition 1 if a rigorous Bernstein race[11] was applied. We consider a rule for choosing the number N of resamplings (it could be something else than Bernstein races), ensuring that the error rate is less than $1 - \delta$.

The error rate is defined as follows. When comparing search points x and x', it is the probability of concluding that $\mathbb{E}f(x, w) > \mathbb{E}f(x', w)$ (or, respectively, $\mathbb{E}f(x', w) > \mathbb{E}f(x, w)$) whereas it is wrong.

Proposition 2 (Races can have infinite loops on plateaus) *Consider $i \in \{1, \dots, \lambda\}$ and $n \geq 1$. Consider $x = p_i$ and $x' = p''_i$ at iteration n of a DE algorithm. Assume that the resampling number N ensures that the error rate is less than $1 - \delta$ for some $\delta > 0$. Then, there is an objective function f and two search points x and x' such that with probability at least $1 - \delta$, N is infinite.*

Proof: Let us assume that the fitness and the search points are such that $f(x, w)$ and $f(x', w)$ have the same probability distribution, with bounded density. Let us assume that the error rate is less than $1 - \delta$ for some $\delta > 0$, and let us show that N is infinite with probability at least $1 - \delta$.

First, due to the bounded density, the probability that $\hat{\mathbb{E}}f(x, w) = \hat{\mathbb{E}}f(x', w)$ is zero. This is because $f(x, w) - f(x', w)$ has a bounded density, hence $\hat{\mathbb{E}}f(x, w) - \hat{\mathbb{E}}f(x', w)$ has a bounded density, hence it is null with probability 0.

Therefore, the resampling rule, if it stops for a finite number N of resamplings, will conclude, with probability 1, that x is better than x', or that x' is better than x. It can not conclude that x and x' have the same expected fitness.

But, by assumption, the only correct answer is that x and x' have the same expected fitness.

Therefore it will conclude erroneously as there is no significant difference to find: the error rate $e \geq P(N < \infty)$.

But, by assumption, the error rate should be $e \leq \delta$: therefore $P(N < \infty) \leq e \leq \delta$. Therefore $P(N = \infty) \geq 1 - \delta$.

■

This explains why upper bounds on numbers of resamplings are necessary when applying Bernstein races. Bernstein races without such a trick can lead to an infinite number of resamplings.

Our rules $N_{adaptive}$ and $N_{enhanced\ adaptive}$ verify the almost sure halting property, which is a good piece of news. However, associated drawbacks (which can't be avoided, by properties above) are

- the resampling loop will eventually stop and conclude that there is a difference whenever there is no significant difference and
- the resampling loop might fail (in terms of detecting the best), with probability arbitrarily close to $\frac{1}{2}$, in case of very close fitness values.

TABLE I: Notation of resampling rules used in the presented experiments (Figures 1, 2, 3 and 4).

Rule	Formula	Notation in the figures
Constant	1	N_{cnst}
Linear	n	N_{lin}
Square root	$\lceil\sqrt{n}\rceil$	N_{sqrt}
Scale	$\lceil\frac{1}{D^2}exp(\frac{4n}{5D})\rceil$	N_{scale}
Adaptive	Eq.2	N_{adap}
Enhanced adaptive	Eq.3	N_{eadap}
Exponential	2^n	$2exp$
Exponential	$\lceil 1.1^n\rceil$	$1.1exp$
Exponential	$\lceil 1.01^n\rceil$	$1.01exp$

The *enhanced* version is pragmatically designed for avoiding uselessly long computations at the early stages. These elements show the difficult compromises when designing resampling rules based on statistical testing. Controlling the misranking probability leads to a risk of infinite resampling loop. When using empirical variance in tests, or when the noise has rare large values, another risk is the underestimation of the noise variance. Rigorous Bernstein races need bounds on possible values (the range parameter which is usually denoted by R in Bernstein races), which are hardly available in practice.

IV. Experiments

Experimental results using different resampling rules are presented in this section. We refer to Table I for a description of the resampling rules and notations used in the presented experiments.

A. The CEC 2005 testbed

The CEC-2005 testbed [27] contains 25 benchmark functions. These functions are grouped into two main categories : *Unimodal functions* and *Multimodal functions*. There are 5 Unimodal functions $F_1, \ldots, F_5$ and 20 Multimodal functions grouped into 3 sub-categories. The first subcategory corresponds to 7 basic functions, named $F_6, \ldots, F_{12}$. The second sub-category are 2 expanded functions F_{13} and F_{14}. The last 11 Multimodal functions $F_{15}, \ldots, F_{25}$ are hybrid composite functions. F_1 is the translated sphere function. F_3 is an ill-conditioned elliptic function. F_5, F_8 and F_{20} have their global optimum on the frontier of the domain.

B. Parameter selection for DE on the CEC 2005 testbed

[28] used a portfolio with competition of parameter settings on DE/best/2 and DE/rand/1. Different F and Cr are used in the mutation and crossover steps with a probability updated at each iteration. A competition between different parameter settings was tested on two unimodal functions (first and second De Jong functions) and four multimodal functions (Rastrigin function, Schwefel function, Griewangk function and Ackley function) in dimension 5, 10 and 30. In this paper, focusing on the extension to the strong noise case, we use parameters tuned in the noise-free case for DE/rand/2 and add a resampling tool as explained in Section III-C.

C. Adding strong noise in the CEC 2005 testbed

As pointed out in [29], troubles in noisy optimization by evolutionary algorithms start when the search points are close enough to the optimum. When the search points are close enough to the optimum, the noise standard deviation is close to the difference between fitness values. Then, convergence becomes difficult. [30], [31] use a multiplicative noise model in which this never occurs because the variance decreases to zero around the optimum. [32] focuses on either multiplicative noise models, or constant noise models with smaller constants than our strong noise requirement. Strong noise models (constant variance, with magnitude significant compared to fitness variations) have been considered both theoretically[33] and experimentally[34], [6]. We work on DE, which is invariant by addition of a constant to the objective values; so we simplify graphs by considering functions with optimum fitness equal to 0, as follows:

$$f_{\substack{index_{CEC05},\\ noise\ free,\\ translated}}(x) = f_{\substack{index_{CEC05},\\ noise\ free,}}(x) - \inf_x f_{\substack{index_{CEC05},\\ noise\ free,}}(x),$$

where the index $index_{CEC05}$ is the function number as defined in the classical CEC 2005 testbed[27]. In this work, we consider the (noise free, translated) CEC 2005 testbed, and create a strongly noisy counterpart as follows:

$$f_{\substack{index_{CEC05},\\ noisy,\\ translated}}(x) = f_{\substack{index_{CEC05},\\ noise\ free,\\ translated}}(x) + f_{\substack{index_{CEC05},\\ noise\ free,\\ translated}}(0) \times \mathcal{N}$$

where $\mathcal{N}$ is a standard Gaussian noise. The standard deviation of the noise is

$$f_{\substack{index_{CEC05},\\ noise\ free,}}(0) - \inf_x f_{\substack{index_{CEC05},\\ noise\ free,}}(0).$$

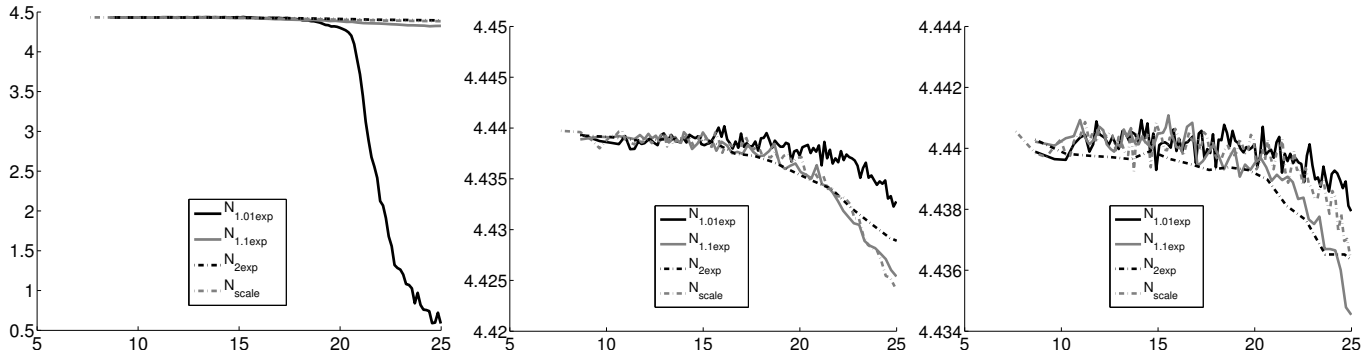
This noise model is easy to reproduce (just adding a Gaussian noise, using the fitness at 0 as standard deviation), and scales with the fitness values, leading to a strong noise model.

D. Experimental results

We do experiments in dimension 2, 10 and 30. Figure 1 presents results on the "CEC 2005+Strong Noise" testbed described in Section IV-C, in dimension 10 (left) and 30 (right), for functions F_{21} to F_{25}. Figure 2 presents results on the "CEC 2005+Strong Noise" testbed described in Section IV-C, in dimension 30, for functions F_{11} to F_{20}. Essentially, N_{lin} is usually the best or among the best; N_{scale} is sometimes better but there is no clear advantage for this more sophisticated formula. Experimental results for functions F_1 to F_{10} are not presented. Basically, F_8, F_9, F_{10} are poorly handled by all algorithms; for F_1-F_7, N_{scale} and N_{lin} perform best. An advantage of N_{scale} is that [10] has proved that exponential rules, for a relevant choice of parameters, is consistent, i.e. guarantees some convergence rates when the original algorithm (without resampling) converges in the noise-free case. Also, the scaling analysis in [25] suggests some values of the parameters. Therefore, we also compared the heuristically derived formula N_{scale} to other exponential rules, $N_{2exp} = 2^n$, $N_{1.1exp} = 1.1^n$ and $N_{1.01exp} = 1.01^n$, and could check that

- the best coefficient in the exponential varies with the dimension;
- the heuristically derived equation N_{scale} approximately finds the right exponent, but some other formulas have approximately the same results.

These results are presented in Figure 4 for functions F_1-F_5.



(a) F8, dimension 2. (b) F8, dimension 10. (c) F8, dimension 30.

Fig. 3: (*x-axis:* $\log_2(number\ of\ evaluations)$, *y-axis:* $\log_2(simple\ regret)$) Test of exponential numbers of resamplings, including the heuristically derived formula "$scale$" and with larger numbers of function evaluations for F_8 (Shifted Rotated Ackleys Function with Global Optimum on Bounds). Standard deviations are tiny and almost invisible. Only the exponential rule with small exponent 1.01 finds a reasonable approximation of the optimum for F_8 in dimension 2. No algorithm finds a reasonable approximation of the optimum in dimension 10 or dimension 30.

V. Discussion & conclusions

Our experiments are performed in the strongly noisy case, i.e. standard deviation of noise approximately equal to the range of fitness values.

A. Main observations

The simple linear resampling method performs well in general. However, the adaptive methods have a similar or better slope at the end, which suggests that they might be better asymptotically - though we have already reached huge numbers of fitness evaluations. The enhanced adaptive method performs better than its non-adaptive counterpart. It just restricts the number of evaluations to 2^n, with n the generation index. The N_{scale} formula performs well and in particular performs always nearly as well as the best of other formulas. However, N_{lin} or $N_{1.01exp}$ perform very similarly.

B. Special cases

F_8 (Shifted Rotated Ackleys Function with Global Optimum on Bounds) and F_{14} (Shifted Rotated Expanded Scaffer's Function (F_6)) are very hard in the strongly noisy case; no algorithm finds a reasonable approximation of the optimum except F_8 in dimension 2 where only the "$scale$" formula succeeds (see Figure 3). On F_{13} (Expanded Extended Griewanks plus Rosenbrocks Function ($F_8 + F_2$)), just one sampling ($r_n = 1$) is enough for the early iterations; but, asymptotically, increasing the number of resamplings becomes necessary in dimension 10 - not in dimension 30 for the considered budget.

C. Conclusion

The contributions of this work are as follows. We propose a strong noise model. This model is more difficult, but close to many real world problems. In many problems, the standard deviation of the noise is close to the differences between fitness values (Section I). The literature proposes various non-adaptive rules, and provides theoretical guarantees, for evolution strategies. We here transfer these rules to differential evolution.

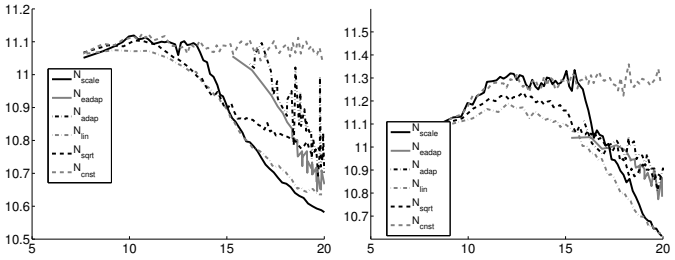
Our rules were able to match the theoretical limit $-\frac{1}{2}$ for a class of algorithms described in [35]. It is not proved that DE is in the scope of [35]. Still, the agreement between Fig. 4 and theory is striking.

We pointed out problems with adaptive rules: difficulties with equal fitness values (e.g. plateaus), leading to possibly infinite loops in case of resamplings until statistically significant differences.

The main take-home messages of this work on resampling numbers in evolution strategies are: (i) $N_{scale} = \lceil D^{-2} \exp(\frac{4n}{5D}) \rceil$, heuristically derived in [25], performs reasonably well but it does not clearly outperform other formulas and in particular the simple and robust $N_{lin} = n$ and other exponential formulas with small coefficients, e.g. $N_{1.01exp} = 1.01^n$. Smaller numbers of resamplings or faster exponentials (e.g. $N_{1.1exp} = 1.1^n$) do not provide satisfactory results. Therefore, we might recommend $N_{1.01exp}$ for example.

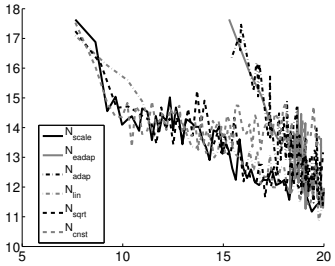
(ii) Adaptive methods might be merged with bounds on resampling numbers (as shown by the compared performance of the enhanced adaptive method, compared to the default adaptive method); in our results non-adaptive methods such as N_{scale} improve adaptive methods by setting a limit on the numbers of resamplings, avoiding wasted evaluations in early stages.

(iii) Non-adaptive bounds added on top of adaptive methods, improve these adaptive methods, but the fact that adaptive methods improve non-adaptive methods is unclear in this setting. The adaptive method do not bring clear improvements compared to simple non-adaptive schemes and were indeed usually worse. This is consistent with e.g. [35] which finds better results for an evolution strategy with a simple non-adaptive rule than with uncertainty handling version such as UH-CMA[26]. Still, these combinations (a good non-adaptive formula plus an adaptive rule based on statistical testing) might be good for easier models of noise when a transient regime in which noise is negligible can be improved by reducing resamplings in the early stages - but on the long run there is no improvement for the additive noise model.

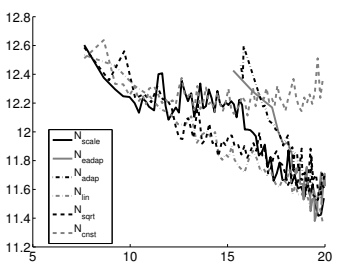


(a) F21, dimension 10.

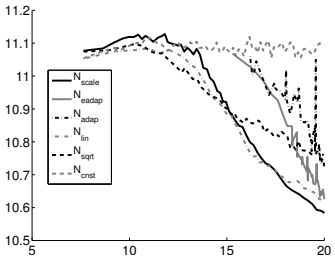
(b) F21, dimension 30.



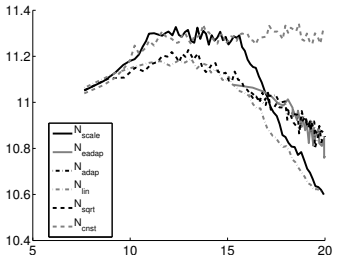
(c) F22, dimension 10.



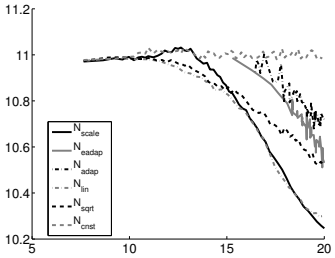
(d) F22, dimension 30.



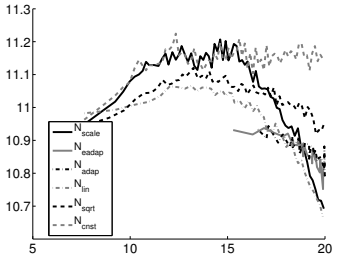
(e) F23, dimension 10.



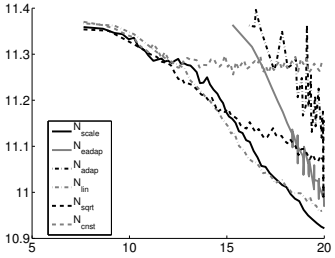
(f) F23, dimension 30.



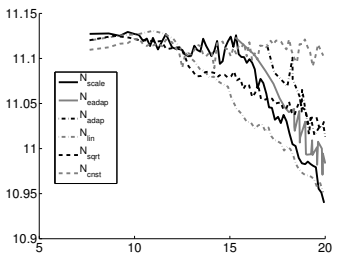
(g) F24, dimension 10.



(h) F24, dimension 30.

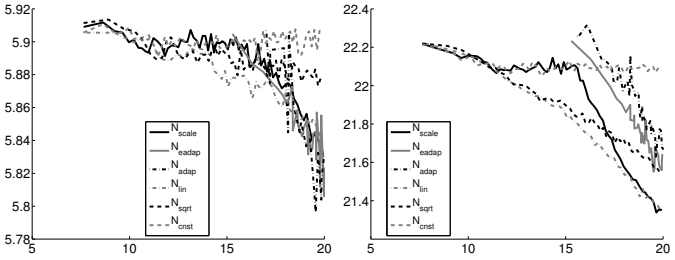


(i) F25, dimension 10.



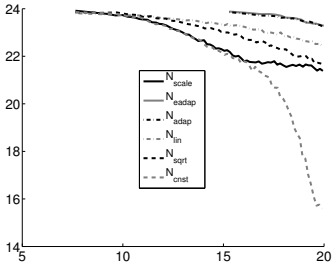
(j) F25, dimension 30.

Fig. 1: (*x-axis:* $\log_2(number\ of\ evaluations)$, *y-axis:* $\log_2(simple\ regret)$) Multimodal functions F_{21} to F_{25} for dimension 10 (left) and dimension 30 (right). Results similar to results for F_1 to F_{20}, i.e. the "$scale$" and the linear formulas dominate. Standard deviations are present but tiny and almost invisible.

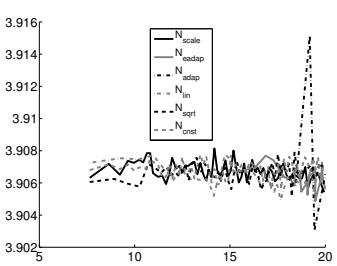


(a) F11, dimension 30.

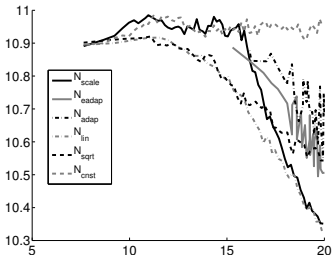
(b) F12, dimension 30.



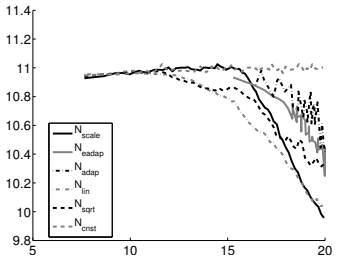
(c) F13, dimension 30.



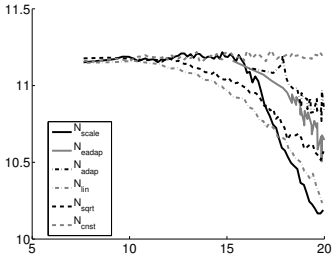
(d) F14, dimension 30.



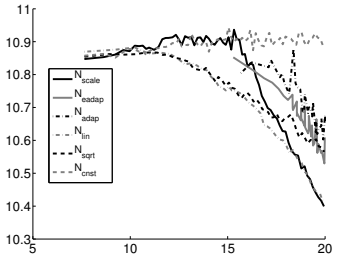
(e) F15, dimension 30.



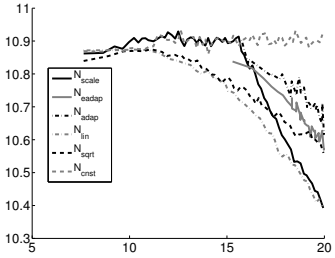
(f) F16, dimension 30.



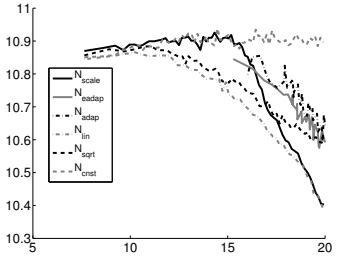
(g) F17, dimension 30.



(h) F18, dimension 30.

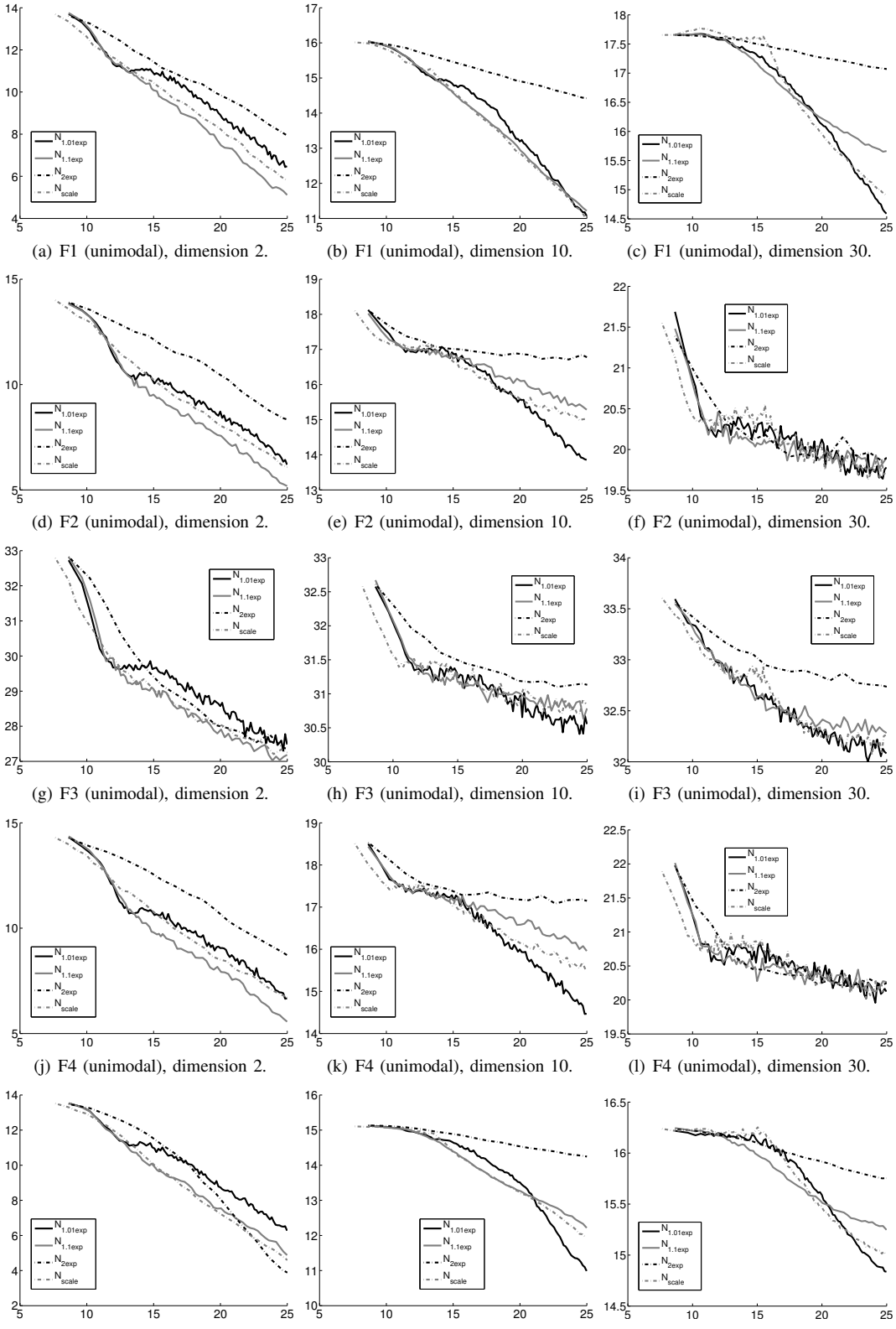


(i) F19, dimension 30.



(j) F20, dimension 30.

Fig. 2: (*x-axis:* $\log_2(number\ of\ evaluations)$, *y-axis:* $\log_2(simple\ regret)$) Multimodal functions F_{11} to F_{20} for dimension 30. Results on F_{13} are quite special; see discussion in Section V-B. Linear and "$scale$" numbers of resamplings dominate. Standard deviations are present but tiny and almost invisible.



(a) F1 (unimodal), dimension 2. (b) F1 (unimodal), dimension 10. (c) F1 (unimodal), dimension 30.

(d) F2 (unimodal), dimension 2. (e) F2 (unimodal), dimension 10. (f) F2 (unimodal), dimension 30.

(g) F3 (unimodal), dimension 2. (h) F3 (unimodal), dimension 10. (i) F3 (unimodal), dimension 30.

(j) F4 (unimodal), dimension 2. (k) F4 (unimodal), dimension 10. (l) F4 (unimodal), dimension 30.

(m) F5 (unimodal), dimension 2. (n) F5 (unimodal), dimension 10. (o) F5 (unimodal), dimension 30.

Fig. 4: (*x-axis:* $\log_2(number\ of\ evaluations)$, *y-axis:* $\log_2(simple\ regret)$) Test of exponential numbers of resamplings, including the heuristically derived formula "$scale$" and with larger numbers of function evaluations. Unimodal functions F_1 to F_5, dimension 2 (left), 10 (middle) and 30 (right) respectively. Standard deviations are tiny and almost invisible. (i) $N_{1.1exp}$ outperforms $N_{1.01exp}$ and N_{2exp} in dimension 2; (ii) $N_{1.01exp}$ is dominant for F_2-F_5 in dimension 10, $N_{1.1exp}$ and N_{scale} perform similarly for F_1 in dimension 10; (iii) $N_{1.01exp}$ is dominant in dimension 30, N_{scale} has sometimes similar performance. In small dimension, the slope in the log-log representation is close to $-\frac{1}{2}$; this is the optimal possible rate for a wide class of algorithms as discussed in [35].

REFERENCES

[1] R. Storn and K. Price, "Differential evolution: A simple and efficient heuristic for global optimization over continuous spaces," *J. of Global Optimization*, vol. 11, no. 4, pp. 341–359, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1023/A:1008202821328

[2] A. Auger, N. Hansen, J. Perez Zerpa, R. Ros, and M. Schoenauer, "Experimental comparisons of derivative free optimization algorithms," in *8th International Symposium on Experimental Algorithms*, no. 5526. Springer Verlag, 2009, pp. 3–15. [Online]. Available: http://hal.inria.fr/inria-00397334/en/

[3] N. Hansen and S. Kern, "Evaluating the cma evolution strategy on multimodal test functions," in *Parallel Problem Solving from Nature - PPSN VIII*, ser. Lecture Notes in Computer Science, X. Y. et al, Ed. Springer Berlin Heidelberg, 2004, vol. 3242, pp. 282–291.

[4] A. LaTorre, S. Muelas, and J.-M. Pena, "Large scale global optimization: Experimental results with mos-based hybrid algorithms," in *Evolutionary Computation (CEC), 2013 IEEE Congress on*. IEEE, 2013, pp. 2742–2749.

[5] S. Das and P. N. Suganthan, "Differential evolution: A survey of the state-of-the-art," *IEEE Trans. on Evolutionary Computation*, vol. 15, no. 1, pp. 4–31, 2011.

[6] R. Coulom, "Clop: Confident local optimization for noisy black-box parameter tuning," in *Advances in Computer Games*. Springer Berlin Heidelberg, 2012, pp. 146–157.

[7] R. S. Sutton and A. G. Barto, *Reinforcement learning: An introduction*. Cambridge, MA: MIT Press., 1998.

[8] 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.

[9] M. Jebalia, A. Auger, and N. Hansen, "Log linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments," *Algorithmica*, 2010. [Online]. Available: http://hal.inria.fr/inria-00433347

[10] S. Astete-Morales, J. Liu, and O. Teytaud, "log-log convergence for noisy optimization," in *Proceedings of EA 2013*, ser. LLNCS. Springer, 2013, p. accepted.

[11] V. Mnih, C. Szepesvári, and J.-Y. Audibert, "Empirical Bernstein stopping," in *ICML '08: Proceedings of the 25th international conference on Machine learning*. New York, NY, USA: ACM, 2008, pp. 672–679.

[12] 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.

[13] J. Brest, S. Greiner, B. Boskovic, M. Mernik, and V. Zumer, "Self-adapting control parameters in differential evolution: A comparative study on numerical benchmark problems," *Trans. Evol. Comp*, vol. 10, no. 6, pp. 646–657, Dec. 2006. [Online]. Available: http://dx.doi.org/10.1109/TEVC.2006.872133

[14] J. Liu and J. Lampinen, "A fuzzy adaptive differential evolution algorithm," *Soft Comput.*, vol. 9, no. 6, pp. 448–462, Jun. 2005. [Online]. Available: http://dx.doi.org/10.1007/s00500-004-0363-x

[15] M. Yang, J. Guan, Z. Cai, and L. Wang, "Self-adapting differential evolution algorithm with chaos random for global numerical optimization," in *Advances in Computation and Intelligence*, ser. Lecture Notes in Computer Science, Z. Cai, C. Hu, Z. Kang, and Y. Liu, Eds. Springer Berlin Heidelberg, 2010, vol. 6382, pp. 112–122. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-16493-4_12

[16] K. V. Price, R. M. Storn, and J. A. Lampinen, *Differential Evolution - A Practical Approach to Global Optimization*, ser. Natural Computing. Springer-Verlag, January 2006, iSBN 540209506.

[17] M. E. H. Pedersen, "Tuning & simplifying heuristical optimization," Ph.D. dissertation, University of Southampton, January 2010. [Online]. Available: http://eprints.soton.ac.uk/342792/

[18] R. Storn, "On the usage of differential evolution for function optimization," in *Fuzzy Information Processing Society, 1996. NAFIPS. 1996 Biennial Conference of the North American*. IEEE, 1996, pp. 519–523.

[19] T. Krink, B. Filipic, G. Fogel, and R. Thomsen, "Noisy optimization problems - a particular challenge for differential evolution?" in *In Proceedings of 2004 Congress on Evolutionary Computation*. IEEE Press, 2004, pp. 332–339.

[20] S. Das, A. Konar, and U. K. Chakraborty, "Improved differential evolution algorithms for handling noisy optimization problems," in *Evolutionary Computation, 2005. The 2005 IEEE Congress on*, vol. 2, 2005, pp. 1691–1698 Vol. 2.

[21] R. Shahryar, T. Hamid R., and S. Magdy M. A., "Opposition-based differential evolution for optimization of noisy problems," in *Evolutionary Computation, 2006. CEC 2006. IEEE Congress on*, 2006, pp. 1865–1872.

[22] ——, "Opposition-based differential evolution," in *Advances in Differential Evolution*, ser. Studies in Computational Intelligence, U. Chakraborty, Ed. Springer Berlin Heidelberg, 2008, vol. 143, pp. 155–171. [Online]. Available: http://dx.doi.org/10.1007/978-3-540-68830-3_6

[23] B. Liu, X. Zhang, and H. Ma, "Hybrid differential evolution for noisy optimization," in *Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on*, 2008, pp. 587–592.

[24] G. Iacca, F. Neri, and E. Mininno, "Noise analysis compact differential evolution," *Intern. J. Syst. Sci.*, vol. 43, no. 7, pp. 1248–1267, Jul. 2012. [Online]. Available: http://dx.doi.org/10.1080/00207721.2011.598964

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

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

[27] P. N. Suganthan, N. Hansen, J. J. Liang, K. Deb, Y. P. Chen, A. Auger, and S. Tiwari, "Problem definitions and evaluation criteria for the cec 2005 special session on real-parameter optimization," Nanyang Technological University, Singapore, Tech. Rep., 2005.

[28] J. Tvrdık, "Competitive differential evolution," in *MENDEL 2006, 12th International Conference on Soft Computing*. Brno: University of Technology, 2006, pp. 7–12.

[29] H.-G. Beyer and S. Finck, "On the performance of evolution strategies on noisy pdqfs:progress rate analysis," in *IEEE Congress on Evolutionary Computation*. IEEE, 2008, pp. 495–502.

[30] D. V. Arnold and H.-G. Beyer, "Efficiency and mutation strength adaptation of the (mu/mui,lambda)-es in a noisy environment," in *Parallel Problem Solving from Nature*, ser. LNCS, M. S. et al., Ed., vol. 1917. springer, 2000, pp. 39–48.

[31] M. Jebalia and A. Auger, "On multiplicative noise models for stochastic search," in *Parallel Problem Solving From Nature*, dortmund Allemagne, 2008. [Online]. Available: http://hal.inria.fr/inria-00287725/en/

[32] A. Auger, S. Finck, N. Hansen, and R. Ros, "BBOB 2009: Comparison Tables of All Algorithms on All Noisy Functions," INRIA, Technical Report RT-0384, Apr. 2010. [Online]. Available: http://hal.inria.fr/inria-00471253

[33] V. Fabian, "Stochastic Approximation of Minima with Improved Asymptotic Speed," *Annals of Mathematical statistics*, vol. 38, pp. 191–200, 1967.

[34] ——, *Stochastic Approximation*, ser. SLP. Department of Statistics and Probability, Michigan State University, 1971. [Online]. Available: http://books.google.fr/books?id=a0aiMQEACAAJ

[35] S. Astete-Morales, M.-L. Cauwet, and O. Teytaud, "Evolution Strategies with Additive Noise: A Convergence Rate Lower Bound," in *Foundations of Genetic Algorithms*, ser. Foundations of Genetic Algorithms, Aberythswyth, United Kingdom, 2015, p. 9. [Online]. Available: https://hal.inria.fr/hal-01077625