Evolution Strategies with Additive Noise: A Convergence Rate Lower Bound

Sandra Astete-Morales, Marie-Liesse Cauwet, Olivier Teytaud

Evolution Strategies with Additive Noise: A Convergence Rate Lower Bound

Sandra Astete-Morales
TAO, Inria, Lri, Umr Cnrs 8623
Bat. 650, Univ. Paris-Sud
91405 Orsay Cedex, France
sandra-cecilia.astete-morales@inria.fr

Marie-Liesse Cauwet
TAO, Inria, Lri, Umr Cnrs 8623
Bat. 650, Univ. Paris-Sud
91405 Orsay Cedex, France
marie-liesse.cauwet@inria.fr

Olivier Teytaud
TAO, Inria, Lri, Umr Cnrs 8623
Bat. 650, Univ. Paris-Sud
91405 Orsay Cedex, France
olivier.teytaud@inria.fr

ABSTRACT

We consider the problem of optimizing functions corrupted with additive noise. It is known that evolutionary algorithms can reach a simple regret $O(1/\sqrt{n})$ within logarithmic factors, when n is the number of function evaluations. We show mathematically that this bound is tight, at least for a wide family of evolution strategies without large mutations.

Categories and Subject Descriptors

G.1.6 [**Optimization**]: Unconstrained optimization

General Terms

Theory

Keywords

Evolution Strategies, Noisy Optimization, Additive Noise

1. INTRODUCTION

Evolutionary Algorithms (EAs) have received an important amount of attention due to their wide applicability in optimization problems. In particular, they show robustness in front of rugged fitness landscapes. This robustness becomes a strong feature of EAs when facing objective functions corrupted by noise.

Black-Box Noisy Optimization. When we only have access to an *approximate* or *noisy* value of the objective function, the problem is termed a *Noisy Optimization Problem*. Additionally, we can consider that the values of the objective function are given by a *black-box* which receives as an input a feasible point and outputs the value of the (noisy) objective function at that point and that is the only information available regarding to the objective function. This will be termed a *Black-box Noisy Optimization Problem* (BBNOP).

Noise models. To analyze the performance of an algorithm in front of a BBNOP, several *noise models* are considered in the literature. These models are appreciated for their simple and natural design. If we let $f(x)$ be the objective function, then the noisy version of it, $f(x,\omega)$, can be defined as any of the following:

$$\begin{aligned}
\textbf{[Additive noise]}\ f(x,\omega) &= f(x) + N(\omega),\\
\textbf{[Multiplicative noise]}\ f(x,\omega) &= f(x) \times (1 + N(\omega))\\
\textbf{[Actuator noise]}\ f(x,\omega) &= f\left(x + N(\omega)\right)
\end{aligned}$$

where $N(\omega)$ is some random variable with $\mathbb{E}_\omega[N(\omega)] = 0$.

Regardless of the noise model considered, EAs are commonly used to find the optimum of (noisy) objective functions. Nonetheless, their versatility pays a price on the convergence rate; EAs are slower than other methods used to solve BBNOP.

In the remaining of the Introduction we describe the characteristics of Evolution Strategies and discuss the convergence rates reached by them both in noisy and noise-free environments. We compare these rates with the strictly better (or faster) convergence rates reached by algorithms that, opposite to Evolution Strategies, sample feasible points far away from the optimum.

1.1 Evolution Strategies

EAs are usually classified depending on certain specific characteristic in some of the stages that define them. For Evolution Strategies (ESs) in the continuous setting considered in the present paper[1], the traditional **mutation** operator creates an *offspring* by taking the *parent* of a generation and adding to it some random perturbation. This random perturbation is usually extracted from a Gaussian distribution. The operator favors smaller mutations and it is defined as:

$$\begin{aligned}
\mathbb{R}^d \to &\qquad \mathbb{R}^d\\
x \mapsto &\, x + \sigma\mathcal{N}(0, C)
\end{aligned}$$

The term σ is denominated *step-size* and the adaptation of it has been subject of study since the creation of ESs. The first achievement in that direction is the 1/5-success-rule [**?**] which is followed by the study of self-adaptation of the step-size using a variation process on it, which gives birth to the so-called SA-ES: *Self-Adaptive Evolution Strategies* [**?**]. Additionally, [**?**] develops a technique where the whole

[1]Some but not all authors consider that ESs are by definition working in the continuous case; anyway the present paper considers continuous domains only.

covariance matrix C is adapted, leading to the CMA-ES algorithm.

For the **selection** stage, ESs generally use *ranking-based* operators. Thus, when we consider BBNOP, the problem in the selection is the *misranking* of individuals. In other words, if we consider individuals x_1 and x_2 and an additive noise model, then due to the noise perturbation we might obtain $f(x_1, \omega_1) > f(x_2, \omega_2)$ whereas actually the real comparison between individuals is the opposite i.e. $f(x_1) < f(x_2)$. To deal with this problem, specific methods have been studied. The techniques include increasing the population size [**?**], using surrogate models [**?**, **?**] and resampling many times per search point [**?**, **?**]. In this context, resampling means that the query to the black-box is repeated several times for a given search point. Afterwards, some statistics of the repeated sample (usually the mean) is used as the objective function value of the point.

1.2 Typical Convergence behaviour

Calls to the black-box might be expensive, so the goal is to minimize the number of queries necessary to find a good approximation of the optimum of the function. In this paper, in order to measure the error between the approximated optimum given by the algorithm and the real optimum of the objective function, we use the *Simple Regret* criterion (definition in Eq. (3)). We are interested in the relationship between the simple regret and the number of iterations: the simple regret has to converge to 0 and with the least amount of iterations possible. To address this analysis we focus on the graph of both variables either in *log-linear* or *log-log* scale (for precise definitions see Eq. (5) and Eq. (6)).

Convergence rate for ESs in the noise-free or small noise case. For ESs, in the case of noise-free optimization, the convergence typically occurs in *log-linear* scale: the logarithm of the simple regret decreases linearly when the number of iterations increases. Such results can be found in [**?**, **?**, **?**, **?**, **?**]. In some cases, the same behavior can be achieved in the noisy case; typically in the case of variance decreasing faster than in the multiplicative model, and if fitness values are averaged over a constant ad-hoc number of resamplings [**?**].

Convergence rate for ESs in the noisy case. The convergence behaviour with additive noise occurs generally in *log-log* scale: the logarithm of the simple regret decreases linearly as a function of the logarithm of the number of iterations [**?**, **?**, **?**, **?**, **?**].
[**?**] shows mathematically that an exponential number of resamplings (w.r.t. number of iterations) or an adaptive number of resamplings (scaling as a polynomial function of the inverse step-size) can both lead to a *log-log* convergence scale in the case of the sphere function with additive noise when using Evolutionary Strategy.

However, in these two previous cases, further information on the value of the convergence slope are not provided in most of these papers.

Convergence rate for algorithms sampling farther away from the optimum. Not only EAs are used in the resolution of BBNOP. Other techniques for the optimization of functions in noisy environments have been explored in the literature. They usually consist in the development of algorithms that sample far away from the optimum in order to approximate the shape of the objective function, using machine learning or finite differences. In this context, [**?**] and [**?**] approximate the tangent of the objective function through a gradient approximated by finite differences, and use it in the optimization process. They both obtain linear convergence in the log-log scale. Even more, [**?**] proves that the convergence occurs with a slope -1 in the case of strongly convex functions, as detailed later. [**?**] proves similar rates asymptotically (on a wide family of functions; tightness of the rates proved in [**?**]) whereas [**?**] proves them non-asymptotically (including the tightness of the rates). A key feature which is common to all these algorithms is that they sample farther from the optimum than ESs.

1.3 Outline of this paper

Section (2) presents the notations used througout the article. Section (3) covers the formalization of algorithms and the main result of the paper. In Section (3.1) we define the optimization algorithms in a general framework. We also discuss the scope of the definition. The section includes as well a definition for the Evolution Strategies family considered in this paper. Section (3.2) is devoted to the enunciation and proof of the main result of this paper: we show a lower bound on the slope of the *log-log* graph, proving that Evolution Strategies family can not reach rates as fast as those reached by algorithms that approximate the shape of the function thanks to samplings far from the optimum. Section (4) shows the empirical verification of the proved results. We present experiments for Evolution Strategies ((1+1)-ES and UHCMAES), covered by the theorem 1, and for the algorithm in [**?**], which is not covered by our results and presents convergence rates strictly faster than ESs. Finally in section (**??**) we discuss the results both theoretically and empirically and we conclude the work.

2. PRELIMINARIES

Consider d a positive integer and a domain $\mathcal{D} \subset \mathbb{R}^d$. Given a function $f : \mathcal{D} \to \mathbb{R}$, the noisy version of it is a stochastic process, also denoted f but defined as $f : \mathcal{D} \times \Omega \to \mathbb{R}, (x, \omega) \mapsto f(x, \omega)$ where ω represents the realization of a random variable over some Ω. Henceforward, $f(x)$ will be the exact value of f in x whereas $f(x, \omega)$ denotes a noisy value of f in x. $f(x, \omega)$ is supposed to be unbiased, i.e. $\mathbb{E}_\omega f(x, \omega) = f(x)$. We assume that x^* is the unknown exact and unique optimum (minimum) of f.

In the present paper, the noise model corresponds to **additive noise**:

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

with $\mathbb{E}_\omega[N(\omega)] = 0$. The noise is then an additive term, independent of x and with *constant variance*, i.e $\forall x \in \mathcal{D}, \forall \omega$, $Var(f(x, \omega)) = Var(N(\omega))$ is constant.

For any positive integer n, x_n denotes the n^{th} search point and $\tilde{x}_n$ denotes the approximation of the optimum that the algorithm proposes after $(n-1)$ function evaluations. We denote y_n the evaluation of the noisy function in x_n. The sequence of search points and their evaluation on the noisy

function is:

$$Z_n = (x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}) \quad (2)$$

The Simple Regret and the Cumulative Regret, SR_n and CR_n respectively, are defined as follows:

$$SR_n = \mathbb{E}[f(\tilde{x}_n, \omega_n)] - \inf_x f(x); \quad (3)$$

$$CR_n = \left(\sum_{i=1}^{n} \mathbb{E}[f(x_i, \omega_i)]\right) - n \inf_x f(x) \quad (4)$$

The rates of convergence are the slopes of the log-linear or log-log graphs; they are defined as follows:

$$\textbf{Log-linear:} \ \limsup_n \frac{\log SR_n}{n} = -\alpha < 0. \quad (5)$$

$$\textbf{Log-log:} \ \limsup_n \frac{\log SR_n}{\log n} = -\alpha < 0, \quad (6)$$

We consider the slope in log-log graph: $-\alpha$. Precisely, we say that the slope $-\alpha$ is verified on a family F of noisy objective functions if α is such that:

$$\forall f \in F, \ \exists C > 0, \ \forall n \in \mathbb{N}, \ SR_n \leq C/n^{\alpha}. \quad (7)$$

This means that several different slopes might be verified.

3. THEORETICAL ANALYSIS

This section consists of the formalization of the algorithms and the main result of the paper. We present on one side the formalization of the concept of *black-box noisy optimization algorithm* (Algorithm 1, section 3.1) and on the other side a formalization of "classical" ES (Definition 1, Section 3.1). Notably, the ESs covered by the formalization of ES in Definition 1 correspond to a wide family but not all of them. The condition on Eq. 10 refers to the evolution of the step-size and the approximation to the optimum. The latter condition holds provably for some ESs, and probably for much more, but not necessarily for e.g. ESs with surrogate models learnt with large mutations [**?**, **?**, **?**]. Also algorithms using several Gaussian distributions simultaneously or multimodal random variables for generating individuals are not covered. Thus, we mainly consider here ESs with one Gaussian distribution which scales roughly as the distance to the optimum.

In (Section 3.2) we prove the theorem that states that the family of evolution strategies described by the formalization converges at best with rate $-1/2$ (tightness comes from [**?**]).

3.1 Formalization of Algorithms

General Optimization Framework

Basically, an optimization algorithm samples some search points, evaluate them and proposes a recommendation (i.e. an approximation of the optimum) from these information. We formalize an optimization algorithm in Algorithm 1.

The procedures `R` and `SP` correspond to the *Recommendation* and *Search Point* stages of the algorithm. `R` outputs a feasible point that stands as the approximate optimum of the respective iteration and `SP` generates new search points to be evaluated. $\mathcal{I}$ represents an the internal state of the algorithm, possibly modified inside `SP`. The sequence Z_n is

Algorithm 1 General Optimization Framework.

Input: s = random seed, p = parameter, $\mathcal{I}$ = initial internal state
  $n \leftarrow 0$
  loop
    $r = rand(s)$
    $\tilde{x}_n = \texttt{R}(Z_n, r, n, p, \mathcal{I})$. ▷ Recommend
    $x_n = \texttt{SP}(Z_n, r, n, p, \mathcal{I})$. ▷ Search
    $y_n = f(x_n, \omega_n)$ ▷ Evaluate f in search point
    $n \leftarrow n + 1$.
  end loop

as defined in Eq. (2) When $n = 0$, Z_n is void and the points x_0 and $\tilde{x}_0$ are initialized depending on the parameters of the algorithm and on the random seed s. Starting from $n = 1$, both the `R` and the `SP` functions return values depending on the results of previous iterations.

The presented framework is in fact very general. First, if we consider algorithms that make use of populations for the optimization process, this characteristic can be simulated in the framework even when apparently the population size in Algorithm 1 is always 1. For example, let us say we want to check if an algorithm that uses a population of size λ (λ-population-based) can fit the framework. Then, an iteration on the λ-population-based algorithm can be "split" into several iterations in the framework, so that the λ individuals can be generated by λ iterations of a population size 1, just by adapting the `R` and `SP` functions.

Second, thanks to r, randomized algorithms are included. We propose different algorithms which match this framework in Section 4. In particular Algorithm 2 can be presented as a explicit example of an algorithm written to fit the framework described by Algorithm 1.

The framework presented encodes black-box algorithms and therefore both evolution strategies and algorithms reaching fast rates as those presented in [**?**, **?**]. Note that there is no restriction regarding to the distance between x_n and $\tilde{x}_n$. In particular, in the case of [**?**, **?**, **?**] the search points and the recommendation points can be far from each other (it is even desirable). On the contrary, ESs have a search point procedure that dictates that x_n should not be very far from the $\tilde{x}_n$.

Perimeter covered by General Optimization Framework

The fact that Algorithm 1 covers usual definitions of black-box optimization algorithms is not completely trivial and the following comments will clarify its scope. In general, a black-box optimization algorithm:

- It uses, as an oracle, the objective function. Since we consider a black-box setting, it has no access to any internal characteristic of the objective function.
- It is a program, written on some formal model of computation (e.g. Turing machine). As a consequence, its state, at any time, is eiter its initial state (in case we are at the first time step) or a function of its internal state and of the results of requests to the oracle.
- Since the algorithm is an algorithm for optimization, it must provide an approximation of the optimum. Such an approximation is termed "recommendation". We

here decide that the approximation of the optimum should not change between two calls to the objective (i.e. oracle) function, or, at least, that our criterion will not depend on other recommendations than the first recommendation provided by the optimization algorithm after some call to the oracle function. The recommendation is also constant, from the initialization until the first call to the objective function.

Therefore, an optimization algorithm is a sequence of internal computations, which modify an internal state. This sequence is sometimes interrupted by a call to the oracle function, or by a change in the recommendation.

We can then rewrite the algorithm, hiding all internal transformations of the internal state $\mathcal{I}$ between two calls to the oracle in some SP function. The algorithm then evaluates the objective function at x_n (call to the oracle). Next, it proposes a new approximation of the optimum; this computation is encoded in R. We have specified that this does not modify $\mathcal{I}$; but the procedure R can be duplicated inside SP, which is allowed to modify the internal state, if necessary, so this is not a loss of generality. The random seed is available for all functions so that there is no limitation around randomization.

We have assumed that the algorithm never spends infinite computation times between two calls to the oracle, and does not stop. We can just decide that in such a case we report the same output for R and the same output for SP.

In fact, our formalism is even more general than black-box optimization *algorithms*, because nothing specifies that SP or R are computable; we do not need any such assumption in our results. The reader unfamiliar with computability can safely assume that we consider computable functions as in the classical RAM [**?**] or Turing [**?**] model.

Simple Evolution Strategies definition

ESs are black-box optimization algorithms and they fit the framework in Algorithm 1. Nonetheless, one important feature that characterizes them is the way to generate search points. Normally, the sampling of new search points is made "around" the recommendation point of the previous generation. This means that the SP procedure is defined by:

$$\texttt{SP}(\mathfrak{Z}_n) = \texttt{R}(\mathfrak{Z}_n) + \sigma(\mathfrak{Z}_n)\Psi(\mathfrak{Z}_n) \tag{8}$$

where, for short, $\mathfrak{Z}_n = (Z_n, r, n, p, \mathcal{I})$. The step-size $\sigma(\mathfrak{Z}_n)$ is usually updated at each generation. $\Psi(\mathfrak{Z}_n)$ is an independent d-dimensional zero-mean random variable, not necessarily Gaussian, with

$$\mathbb{E}\|\Psi(\mathfrak{Z}_n)\|^2 = d. \tag{9}$$

Also, we consider that the ESs should satisfy the following condition on the evolution of the step-size with regards to the recommendation points:

$$\exists D > 0,\ \forall n \geq 0,\ \mathbb{E}[\sigma(\mathfrak{Z}_n)^2] \leq D\mathbb{E}[\|\tilde{x}_n - x^*\|^2]. \tag{10}$$

Now we can state the definition of ES covered by the theorem in section 3.2

DEFINITION 1. [**Simple Evolution Strategy**] *We define a simple Evolution Strategy as an algorithm that matches framework of Alg. 1 and satisfies both Eq. 8 and Eq. 10.*

Perimeter covered by Simple Evolution Strategy definition

Let us discuss the assumptions in our evolution strategy framework above.

Eq. 9 is not a strong constraint, as one can always rephrase the algorithm for moving multiplicative factors from $\Psi(\mathfrak{Z}_n)$ to $\sigma(\mathfrak{Z}_n)$ so that $\mathbb{E}\|\Psi(\mathfrak{Z}_n)\|^2 = d$.

The assumption in Eq. 8 is easy to understand. It is verified for a classical evolutionary algorithm with a single parent or a μ/μ recombination (i.e. parent equal to the average of selected offspring), including weighted recombinations.

The assumption in Eq. 10 is more difficult to grasp. It means that $\tilde{x}_n$ and $\sigma(\mathfrak{Z}_n)$ decrease at the same rate towards the optimum. The literature provides the following cases:

- The scale-invariant algorithm obviously verifies the assumption, by definition. The scale-invariant algorithm is however an essentially theoretical algorithm, used for theoretical proofs rather than for real applications.
- Related results are proved for some evolutionary algorithms in the noise-free case, as shown in [**?**]; $\sigma(\mathfrak{Z}_n)/\|\tilde{x}_n - x^*\|$ converges to some distribution. [**?**] shows that it is also true for some provably convergent noisy optimization evolutionary algorithm with resamplings. However, it is not clear that these results imply Eq. 10.
- Beyond mathematical proofs (indeed there are many evolutionary algorithms for which we have no convergence proof at all), Eq. 10 is widely verified in experimental results when algorithms converge, in the (1+1)-evolution strategy [**?**], in self-adaptive algorithms [**?**], in Covariance Matrix Adaptations variants [**?**], and indeed most evolutionary algorithms [**?**].

What would be an evolutionary algorithm which does *not* verify Eq. 10 ? A natural example is an evolutionary algorithm which samples far from the current estimate $\tilde{x}_n$ of the optimum, e.g. for building a surrogate model. Interestingly, all optimization algorithms which are fast in noisy optimization with constant noise variance in the vicinity of the optimum verify such a property, namely sampling far from the optimum [**?**, **?**, **?**]. This suggests that modified ESs which include samplings far from the optimum, might be faster.

3.2 Lower bound for Simple Evolution Strategies

We now state our main theorem, namely the proof that evolution strategies, in their usual setting without mutations far from the optimum estimates, can not reach rate as good as algorithms without such restrictions.

THEOREM 1. *Let F be the set of quadratic functions $f : \mathcal{D} \to \mathbb{R}$ defined on $\mathcal{D} = \mathbb{R}^d$ by $f(x) = \frac{1}{2}\|x\|^2 - (x^* \cdot x)$ for some $\|x^*\| \leq \frac{1}{2}$. Consider a simple Evolution Strategy as in definition 1 and the noisy optimization of $f \in F$ corrupted by some additive noise: $f(x, \omega) = f(x) + N(\omega)$ such that $\mathbb{E}_\omega[f(x, \omega)] = f(x)$. Then, for all $\alpha > \frac{1}{2}$, the slope $-\alpha$ is not reached.*

REMARK 1. **(Tightness in the framework of evolution strategies.)** *[?] shows that, within logarithmic factors, an evolution strategy with Bernstein races (with modified sampling in order to avoid huge numbers of resamplings due to individuals with almost equal fitness values) can reach a slope $-\alpha$ arbitrarily close to $-\alpha = -\frac{1}{2}$. To the best of our knowledge, it is not known whether we can reach $\alpha = \frac{1}{2}$.*

Proof: Let us assume, in order to get a contradiction, that a slope $\alpha > \frac{1}{2}$ is reached. Then, $SR_n \leq C/n^\alpha$ for some $\alpha > 1/2$ and $C > 0$.

Notations are similar to Section 3.1: for any $i \in \{1, \ldots, n\}$, $x_i = \texttt{SP}(\mathfrak{Z}_i)$, $\tilde{x}_i = \texttt{R}(\mathfrak{Z}_i)$, $\sigma_i = \sigma(\mathfrak{Z}_i)$, $\Psi_i = \Psi(\mathfrak{Z}_i)$, where Ψ_i are centered independent random variables in $\mathbb{R}^d$ with $\mathbb{E}\|\Psi_i\|^2 = d$. Let us evaluate the cumulative regret:

$$
\begin{aligned}
2CR_n &= 2\sum_{i=1}^{n} (\mathbb{E}f(x_i) - f(x^*)) \text{ by definition in Eq. 4.} \\
&= \sum_{i=1}^{n} \mathbb{E}[\|x_i\|^2 - 2(x^* \cdot x_i) + \|x^*\|^2] \\
&= \sum_{i=1}^{n} \mathbb{E}[\|x_i - x^*\|^2] \\
&= \sum_{i=1}^{n} \mathbb{E}[\|\tilde{x}_i - x^* + \sigma_i \Psi_i\|^2] \text{ by Eq. 8} \\
&\leq \sum_{i=1}^{n} \left(\mathbb{E}\|\tilde{x}_i - x^*\|^2 + \mathbb{E}\sigma_i^2 \mathbb{E}\Psi_i^2\right) \text{ by independence} \\
&\leq \sum_{i=1}^{n} \left(\mathbb{E}\|\tilde{x}_i - x^*\|^2 + d\mathbb{E}\sigma_i^2\right) \text{ by Eq. 9} \\
&\leq 2(1 + dD) \sum_{i=1}^{n} \mathbb{E}[SR_i] \text{ by Eq. 10}
\end{aligned}
$$

The last equation leads to

$$CR_n \leq C(1 + dD)n^{1-\alpha} \tag{11}$$

[?, Theorem 6] has shown that, for any optimization algorithm as defined in Section 3.1, there is at least one function in $f \in F$ for which the cumulative regret is $CR_n \geq 0.02 \min(1, d\sqrt{n})$, which contradicts Eq. 11. □

4. EXPERIMENTAL VERIFICATION OF THE LOWER BOUND

This section is devoted to the verification of the lower bound on the convergence rate for ESs stated in Theorem 1 and the comparison with the convergence rate of a "fast" Algorithm: SHAMIR ALGORITHM [?].

We here show experimentally that the rate -1 promised by [?] is visible on experiments, even with moderate budgets in terms of numbers of function evaluations. We then show that, consistently with theory, we could not do better than slope $-\frac{1}{2}$ with evolution strategies (Section 4.2). All the results in all this section, consider an approximation of the slope of single regret (Eq. (3)) as follows[2]:

$$\log(SR_n/d)/\log(n)$$

4.1 Fast Convergence: Shamir Algorithm

Experiments SHAMIR ALGORITHM

[?] designed an optimization algorithm using stochastic approximation methods ([?, ?]). At each iteration, it computes a natural gradient and uses it to update the estimate of the optimum. This algorithm is described in Algo. 2 and is named SHAMIR ALGORITHM in the following. We point out the fact that SHAMIR ALGORITHM can sample some search points far from the current approximation. This algorithm may theoretically reach some slope $\alpha = 1$. Importantly, we present the algorithm in the framework of Section 3.1, however, neither Eq. 8, nor Eq. 10 are satisfied so that this cannot be considered a Simple Evolution Strategy as in Def. (1).

Algorithm 2 SHAMIR ALGORITHM. Written in the general optimization framework.

procedure $\texttt{R}(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}, r, n, p, \mathcal{I})$
  ▷ $\mathcal{I}$ is a vector of n elements in the domain.
  if $\|\mathcal{I}_n\| \geq B$ **then**
    $\mathcal{I}_n = B\frac{\mathcal{I}_n}{\|\mathcal{I}_n\|}$
  end if
  $\tilde{x}_n = \frac{2}{n}\sum_{j=\lceil n/2 \rceil, \ldots, n} \mathcal{I}_j$
end procedure
procedure $\texttt{SP}(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}, r, n, p, \mathcal{I})$
  if $n = 0$ **then**
    $\mathcal{I} = (0)$
    Return $x_0 = 0$
  end if
  Compute $x_n = x_{n-1} + \frac{\epsilon}{\sqrt{d}} r$
  Compute $\tilde{g} = \frac{\sqrt{d} y_{n-1}}{\epsilon} r$
  Compute $\mathcal{I} = (\mathcal{I}, x_{n-1} - \frac{1}{\lambda n}\tilde{g})$
end procedure

Input: : $p = (\lambda, \epsilon, B) \in \mathbb{R}_+ \times (0, 1] \times R_+$, $s = random\ seed$
$n \leftarrow 0$
loop
  Generate $r \in \{-1, 1\}^d$, uniformly and randomly
  $\tilde{x}_n = \texttt{R}(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}, r, n, p, \mathcal{I})$
  $x_n = \texttt{SP}(x_0, \ldots, x_{n-1}, y_0, \ldots, y_{n-1}, r, n, p, \mathcal{I})$
  $y_n = f(x_n)$
  $n \leftarrow n + 1$
end loop

We compare the performance of Shamir's Algorithm on the noisy sphere function: $x \mapsto \|x - 0.5\|^2 + \mathcal{N}(0, 1)$, where $\mathcal{N}(0, 1)$ is a standard Gaussian.

Results are presented in Figure 1. Experiments are performed in various dimensions: 2, 4, 8 and 16. We observe that independently of the dimension, the algorithm's slope is smaller than $-1/2$ (i.e. faster) and converges toward -1 (i.e. faster than the bound we have proved).

[2]Note that dividing by d does not matter asymptotically and both theory [?] and experiments show that it is a good normalization for convergence rates.

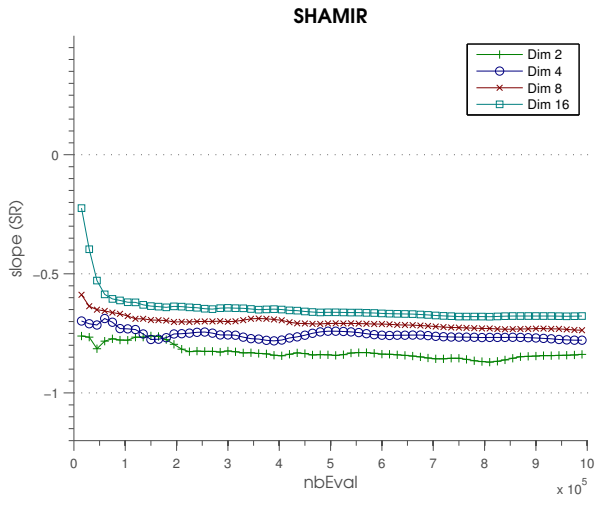


Figure 1: Shamir's algorithm [**?**] on the sphere function $x \mapsto \|x\|^2 + \mathcal{N}(0,1)$ where $\mathcal{N}(0,1)$ is an independent Gaussian standard random variable. X-axis: number of function evaluations. Y-axis: estimate of the slope (see Equation 4). The maximum standard deviation for all averages presented here (experiments are averaged over 21 runs) is 10^{-3}.

4.2 Slow Convergence: UHCMAES and 1+1 ES

Experiments UHCMAES

UH-CMA-ES (Uncertainty Handling Covariance Matrix Adaptation Evolution Strategy) was introduced in [**?**]. This algorithm is a specific variant of CMA-ES designed for dealing with noise. More specifically, it uses an adaptive number of resamplings in order to reduce the noise. It combines the traditional CMA-ES algorithm with an Uncertainty-Handling tool. The Uncertainty-Handling tool is made of two parts. The first part *measures* the uncertainty due to the noise and the second part *handles* the uncertainty. The treatment of the uncertainty is twofold. If the measurement of the uncertainty exceeds a given threshold, then *the computation time* (typically the number of resamplings) increases and/or the variance (the step-size) of the population increases. Whereas if the uncertainty is below the threshold, the *computation time* decreases.

The notion of increased *computation time* varies from a problem to another. Noise may come from inaccurate values given by the captors. In this case, attributing more time to a captor might improve its precision. In case of Monte Carlo estimates, reproducing Monte Carlo runs and averaging the results with decrease the variance. This second method, i.e. resampling, is the one we are interested in; therefore the *computation time* refers to the number of resamplings in this paper.

We provides a high-level pseudo-code of UH-CMA-ES in Algo. 3. For the sake of clarity, the pseudo-code given in Algo. 3 is not cast into the setting of Section 3.1. However, as any optimization algorithm, it could be rewritten so that it matches the general setting of Algo. 1 (see Section 3.1).

The CMA-ES part of the algorithm generates a new population at each iteration. The new population is obtained by mutating the old one thanks to a Gaussian random variable. The mean, the variance σ^2 and a (scaled) covariance C of this Gaussian random variable are adapted at each iteration, depending of the selection/ranking of the μ best offspring. Then, the center of the Gaussian is recommended as an approximation of the optimum. The so-called evolution path p_σ (resp. p_C) of σ (resp. C) is updated and used to update σ (resp. C). All these updates are grouped into Line 10 and are based on both the old and new population, parameters σ, p_σ, C, p_C and parameters which are not detailed. For brevity, these updates are not detailed in Alg. 3, see [**?**] for extra information.

The "UH" part is based on the resamplings (Line 5), on the evaluation of the uncertainty (**Generate** λ' and **Compute** threshold $\bar{t}$) and consequently, on the adjustment of parameters σ and r. The two procedures **Generate** λ' and **Compute** threshold $\bar{t}$ are described in Subroutines **??** and **??** respectively.

Algorithm 3 UH-CMA-ES. $\mathcal{N}(a,b)$ stands for a normal random variable of mean a and covariance b. *hparam* stand for hidden parameters, it includes the parameters used in the update of σ, p_σ, C, p_C, in functions **Generate** λ' and **Compute threshold** $\bar{t}$.

Require: $\lambda \in \mathbb{N}$, $\alpha_r \in \mathbb{R}$, $\alpha_\sigma \in \mathbb{R}$, *hparam*
1: **Initialization:** $x_i = 0 \in \mathbb{R}^d$, $\forall i \in \{1,\dots,\lambda\}$, $m = 0$, $\sigma = 0.6$, $C = I$, p_σ, p_c, $r = 1$, $\epsilon = 10^{-7}$, $\bar{t} = 0$
2: **while** not terminate **do**
3: **for** $i = 1$ *to* λ **do**
4: $x_i \leftarrow \mathcal{N}(m, \sigma^2 C)$
5: $y_i \leftarrow \frac{1}{r}\sum_{j=1}^{r} (f(x_i) + noise_\omega)$
6: **end for**
7: Sort (y_i) such that $y_{s(1)} \le \cdots \le y_{s(\lambda)}$
8: $(x_1,\dots,x_\lambda) \leftarrow (x_{s(1)},\dots,x_{s(\lambda)})$
9: $m \leftarrow \frac{1}{\lambda}\sum_{i=1}^{\lambda} x_i$
10: Update parameters σ, C, p_σ, p_c
11: **Generate** λ' ▷ possibly 0
12: **for** $i_{'}= 1$ *to* λ' **do**
13: $x_i^{'} \leftarrow x_i + \epsilon\sigma\mathcal{N}(0, C)$
14: $y_i^{'} \leftarrow \frac{1}{r}\sum_{j=1}^{r} \left(f(x_i^{'}) + noise_\omega\right)$
15: **end for**
16: **Compute threshold** $\bar{t}$.
17: $y_i^{''} \leftarrow \frac{y_i + y_i^{'}}{2}$ if $i \le \lambda'$ and $y_i^{''} \leftarrow y_i$ otherwise
18: Sort $(y_i^{''})$ such that $y_{s^{''}(1)}^{''} \le \cdots \le y_{s^{''}(\lambda)}^{''}$
19: $(x_1,\dots,x_\lambda) \leftarrow (x_{s(1)},\dots,x_{s(\lambda)})$
20: **if** $\bar{t} > 0$ **then**
21: $r \leftarrow \alpha_r r$, $\sigma \leftarrow \alpha_\sigma \sigma$
22: **else**
23: $r \leftarrow \frac{r}{\alpha_r^{0.25}}$
24: **end if**
25: **end while**
26: **return:** x_1

We here experiment this algorithm on $f(x) = \|x - x^*\|^2 + 0.3\mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ is a standard Gaussian random variable and with $x^* = 0.5$. In these experiments, we set $r = \alpha_r = 1$, hence we have only one evaluation per point. The only way to deal with noise is the one by default [3] to increase the step-size σ.

[3] Other settings by default in the implementation are found at URL `https://www.lri.fr/~hansen/cmaes.m`.

Subroutine 1 Generate $\lambda^{'}$

Input: $g_\lambda = \max(0.2, \frac{2}{\lambda})$
if $\lambda^{'} = 0$ for more then $\frac{2}{g_\lambda \lambda}$ generations **then**
  $\lambda^{'} \leftarrow 1$
else
  $\lambda^{'} \leftarrow \lfloor g_\lambda \times \lambda \rfloor + 1$ with probability $g_\lambda \times \lambda - \lfloor g_\lambda \times \lambda \rfloor$
  $\lambda^{'} \leftarrow \lfloor g_\lambda \times \lambda \rfloor$ otherwise
end if
return $\lambda^{'}$

Subroutine 2 Compute threshold $\bar{t}$. *sgn* means 'sign'. $\Delta_\theta^{lim}(R)$ is the $\theta \times 50\%$-ile of the set $\{|1-R|, |2-R|, \ldots, |2\lambda - 1 - R|\}$.

Input: $\theta = 0.2$, $c_t = 1$, (y_i), $(y_i^{'})$
Rank $Y = (y_i) \cup (y_i^{'})$
for $i \in \{1, \ldots, \lambda^{'}\}$ **do**
  $\Delta_i \leftarrow rank(y_i^{'}) - rank(y_i) - sgn(rank(y_i^{'}) - rank(y_i))$
end for
Compute

$$t \leftarrow \frac{1}{\lambda^{'}} \sum_{i=1}^{\lambda^{'}} \left(2|\Delta_i| - \Delta_\theta^{lim} \left(rank(y_i^{'}) - \mathbf{1}_{y_i^{'} > y_i} \right) - \Delta_\theta^{lim} \left(rank(y_i) - \mathbf{1}_{y_i > y_i^{'}} \right) \right)$$

$\bar{t} \leftarrow (1 - c_t)\bar{t} + c_t t$
return $\bar{t}$

Results are shown in Figure **??**. Independently of the dimension, convergence to the optimum occurs at the same rate. This rate is approximately $-0.2 > -0.5$.

Experiments $(1+1)$-ES

We consider a simple evolution strategy, namely the $(1+1)$ evolution strategy with one-fifth rule [**?**, **?**], with additional revaluations, implemented as shown in Algorithm **??**.

Algorithm 4 $(1+1)-ES$ for noisy optimization with resamplings. $\mathcal{N}(0,1)$ is a standard Gaussian. The function *number_of_revaluations* depends on the current iteration and on a parameter p. Typically the number of revaluations is polynomial: n^p or exponential: p^n.

Initialization: $n = 0$, $\sigma = 1$, $x = (0, \ldots, 0)$, p
while not terminate **do**
  $x' \leftarrow x + \sigma \mathcal{N}(0,1)$
  $n \leftarrow n + 1$
  $r \leftarrow number\_of\_revaluations(n, p)$
  $y \leftarrow \frac{1}{r} \sum_{i=1}^{r} (f(x), \omega))$
  $y' \leftarrow \frac{1}{r} \sum_{i=1}^{r} (f(x', \omega))$
  if $y' < y$ **then**
    $x \leftarrow x'$ and $\sigma \leftarrow 2\sigma$
  else
    $\sigma \leftarrow .84\sigma$
  end if
end while
return x

A way to slightly improve Algorithm **??** is to improve the computation of the fitness value of the current best recommendation by averaging the current estimate with the previous estimates of the same search point, when the mutation has not been accepted and $x_n = x_{n-1}$. We propose such modification in Algorithm **??** by using a weighted average in the estimate fitness value of the current best search point.

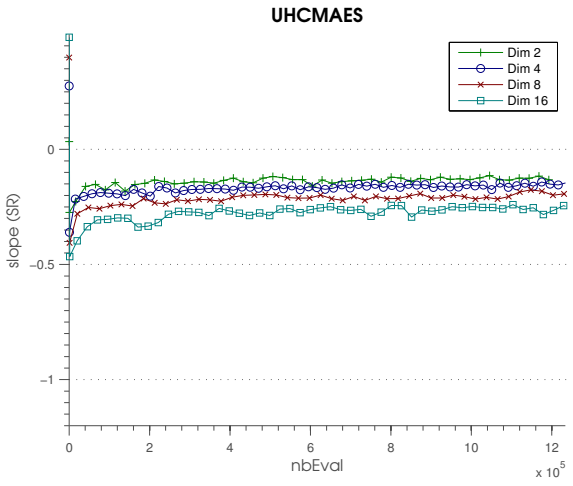


Figure 2: UH-CMA-ES algorithm [**?**] on the sphere function $x \mapsto ||x||^2 + 0.3\mathcal{N}(0,1)$ where $\mathcal{N}(0,1)$ is an independent Gaussian standard random variable. The maximum standard deviation for all averages presented here (experiments are averaged over 21 runs) is 1.

Algorithm 5 Slightly improved $(1+1)-ES$ for noisy optimization with resamplings.

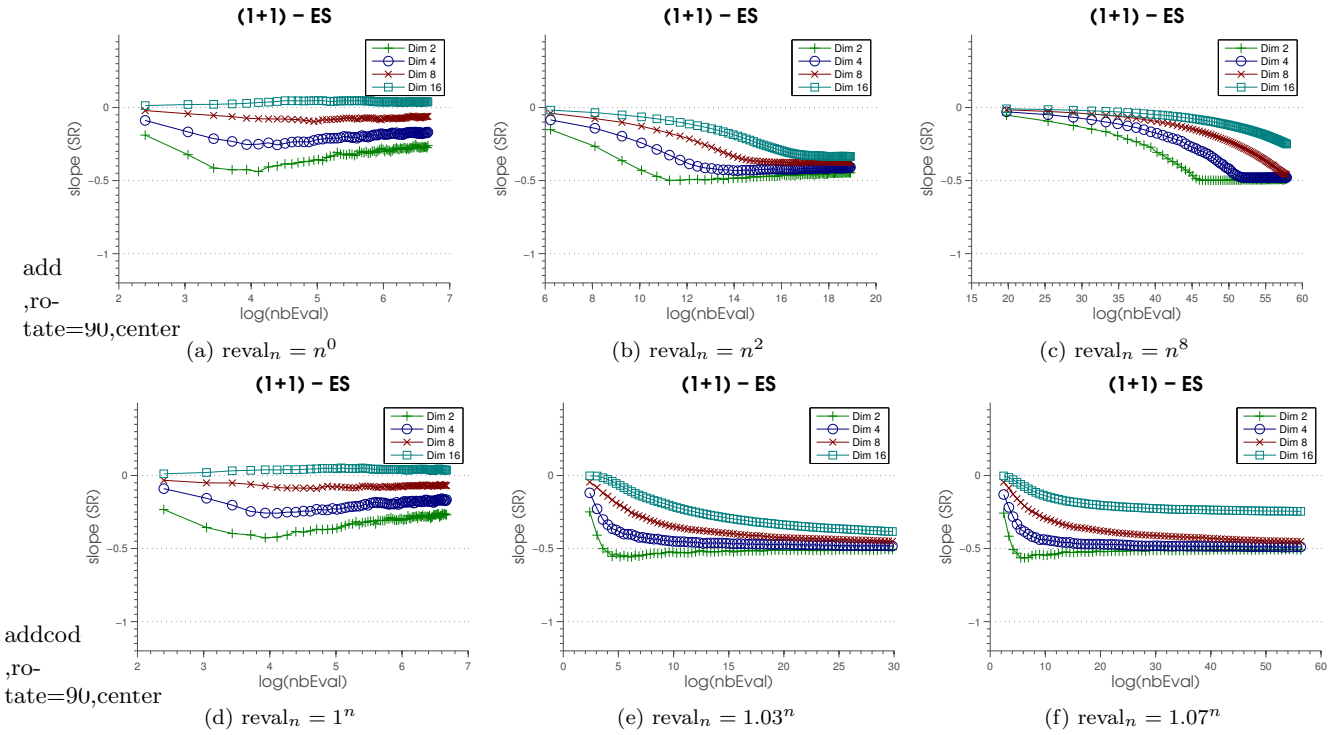
Initialization: $n = 0$, $\sigma = 1$, $k = 0$, $x = (0, \ldots, 0)$, $y = 0$
while not terminate **do**
  $x' \leftarrow x + \sigma \mathcal{N}$ ▷ Gaussian mutation
  $n \leftarrow n + 1$
  $r \leftarrow number\_of\_revaluations(n)$
  $y \leftarrow k \times y + r \times \frac{1}{r} \sum_{i=1}^{r} f(x, \omega)$
  $k \leftarrow k + r$
  $y \leftarrow y/k$
  $y' \leftarrow \frac{1}{r} \sum_{i=1}^{r} f(x', \omega)$
  if $y' < y$ **then**
    $x \leftarrow x'$
    $\sigma \leftarrow 2\sigma$
    $y \leftarrow 0$
    $k \leftarrow 0$
  else
    $\sigma \leftarrow .84\sigma$
  end if
end while

Experiments on the noisy sphere function $\|x - x^*\|^2 + \mathcal{N}(0,1)$, where $\mathcal{N}(0,1)$ is a standard Gaussian, are provided, using Algorithm **??**. Results are presented in Figure **??** in various dimensions (2, 4, 8, 16 respectively). Seemingly both exponential and polynomial resamplings lead to a slope $-1/2$.

5. CONCLUSIONS

Theoretical Results

We have shown that Evolution Strategies, at least under their most common form, can not reach the same rate as noisy optimization algorithms which use evaluations of the objective function farther from the approximate optimum in order to obtain extra information on the function. On the contrary, ESs use the evaluation of objective functions exclusively to determine the ranking of the feasible points in each iteration. Therefore, usual ESs cannot reach rates as the ones in [**?**, **?**]. The latter type of algorithms reach a slope -1, whereas we have a limit at $-1/2$ with evolution strategies. This solves the conjecture proposed in [**?**] just after theorem 1.



(a) $\text{reval}_n = n^0$ (b) $\text{reval}_n = n^2$ (c) $\text{reval}_n = n^8$

(d) $\text{reval}_n = 1^n$ (e) $\text{reval}_n = 1.03^n$ (f) $\text{reval}_n = 1.07^n$

Figure 3: Results 1+1 ES for dimension 2, 4, 8 and 16. First row of plots presents Polynomial resampling and the second row Exponential resampling. The maximum standard deviation for all averages presented here (experiments are averaged over 400 runs) is 0.025.

It is important to note that the result in this paper indeed covers not only Evolutionary Algorithms, but also, for example, many pattern search methods. We proved the results for algorithms which perform sampling within some distance of the approximate optimum, this distance scaling roughly as the distance to the optimum. This property is known to be satisfied by most ESs (see Section 3.1). However, for many algorithms it is verified only experimentally and not formally proved.

ESs with surrogate models are not concerned by our lower bound. More precisely, if we include strong surrogate modelling with large mutations (and so contradicting Eq. 10), then we can recover fast rates with slope -1. An extreme example of this situation is the case in which the sampling/ surrogate model is exactly the algorithm in [**?**], [**?**] or [**?**]. Using them as to obtain surrogate models within an ES will ensure a fast convergence rate for the ES. Obviously, it is desirable to verify if such result can also be obtained with more "evolutionary" approaches.

The bound presented in this paper does not cover evolutionary algorithms that would use very large mutations. Maybe this is a good path to follow for designing fast evolutionary noisy optimization algorithms.

Experimental results

For all experiments we check convergence rates on the sphere function with additive noise.
We consider one algorithm with theoretical fast rate, Shamir Algorithm, and two ESs: UH-CMA-ES and (1+1) ES. For Shamir Algorithm we have achieved a successful implementation ([**?**] delivers the theoretical analysis of his algorithm) and confirmed empirically the fast convergence rate proved in [**?**, **?**] (i.e. slope of SR $= -1$). For UH-CMA-ES and (1+1) ES we have shown that ESs can approximate slope of SR -0.5 using (1+1)-ES. UH-CMA-ES also reaches linear convergence in the log-log scale but with a slower rate (slope of SR around -0.2).

Further work

A first further work consists in proving the result in a wider setting, this is, weakening the assumption in Eq. 10. Another further work is investigating which optimization algorithms, other than evolution strategies, are concerned by our result or by similar results. In the case of strongly convex functions with a lower bound on eigenvalues of the Hessian, we conjecture that the asymptotic rate -1 can also not be reached by the considered family of evolutionary algorithms.

Acknowledgements

Possibilities of algorithms using huge mutations were discussed in the noisy optimization working group at Dagstuhl's seminar this year; we are grateful to N. Hansen, Y. Akimoto, J. Shapiro, A. Prügel-Benett for fruitful discussions there.

6. REFERENCES

[1] D. Arnold and H.-G. Beyer. Investigation of the (μ, λ)-es in the presence of noise. In *Proc. of the IEEE*

Conference on Evolutionary Computation (CEC 2001), pages 332–339. IEEE, 2001.

[2] S. Astete-Morales, J. Liu, and O. Teytaud. log-log convergence for noisy optimization. In *Proceedings of EA 2013*, LLNCS, page accepted. Springer, 2013.

[3] A. Auger. Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible Markov chains. *Theoretical Computer Science*, 334(1-3):35–69, 2005.

[4] A. Auger, M. Jebalia, and O. Teytaud. (x,sigma,eta) : quasi-random mutations for evolution strategies. In *EA*, page 12p., 2005.

[5] H.-G. Beyer. *The Theory of Evolution Strategies*. Natural Computing Series. Springer, Heidelberg, 2001.

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

[7] R. Coulom, P. Rolet, N. Sokolovska, and O. Teytaud. Handling expensive optimization with large noise. In *Foundations of Genetic Algorithms*, 2011.

[8] V. Fabian. Stochastic Approximation of Minima with Improved Asymptotic Speed. *Annals of Mathematical statistics*, 38:191–200, 1967.

[9] 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*, pages 401–408, New York, NY, USA, 2009. ACM.

[10] M. Jebalia, A. Auger, and N. Hansen. Log linear convergence and divergence of the scale-invariant (1+1)-ES in noisy environments. *Algorithmica*, 2010.

[11] I. Rechenberg. *Evolutionstrategie: Optimierung Technischer Systeme nach Prinzipien des Biologischen Evolution*. Fromman-Holzboog Verlag, Stuttgart, 1973.

[12] O. Shamir. On the complexity of bandit and derivative-free stochastic convex optimization. *CoRR*, abs/1209.2388, 2012.

[13] O. Teytaud and H. Fournier. Lower bounds for evolution strategies using vc-dimension. In G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni, and N. Beume, editors, *PPSN*, volume 5199 of *Lecture Notes in Computer Science*, pages 102–111. Springer, 2008.