HAL
open science

Analysis of Different Types of Regret in Continuous Noisy Optimization

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

To cite this version:

Sandra Astete-Morales, Marie-Liesse Cauwet, Olivier Teytaud. Analysis of Different Types of Regret in Continuous Noisy Optimization. Genetic and Evolutionary Computation Conference 2016, Jul 2016, Denver, United States. hal-01347814v1

HAL Id: hal-01347814

https://hal.science/hal-01347814v1

Submitted on 21 Jul 2016 (v1), last revised 23 Jul 2016 (v2)

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Analysis of Different Types of Regret in Continuous Noisy Optimization

Sandra Astete-Morales, Marie-Liesse Cauwet, Olivier Teytaud
TAO/Inria Saclay-IDF, Univ. Paris-Saclay
Bat. 660, rue Noetzlin, Gif-Sur-Yvette, France

July 21, 2016

Abstract

The performance measure of an algorithm is a crucial part of its analysis. The performance can be determined by the study on the convergence rate of the algorithm in question. It is necessary to study some (hopefully convergent) sequence that will measure how "good" is the approximated optimum compared to the real optimum.

The concept of *Regret* is widely used in the bandit literature for assessing the performance of an algorithm. The same concept is also used in the framework of optimization algorithms, sometimes under other names or without a specific name. And the numerical evaluation of convergence rate of noisy algorithms often involves *approximations* of regrets. We discuss here two types of approximations of Simple Regret used in practice for the evaluation of algorithms for noisy optimization. We use specific algorithms of different nature and the noisy sphere function to show the following results. The approximation of Simple Regret, termed here Approximate Simple Regret, used in some optimization testbeds, fails to estimate the Simple Regret convergence rate. We also discuss a recent new approximation of Simple Regret, that we term Robust Simple Regret, and show its advantages and disadvantages.

1 Introduction

The performance measure of an algorithm involves the evaluation of the quality of the approximated optimum with regards to the real optimum. This can be done by using the concept of *Regret*, well studied in the machine learning literature and used in the noisy optimization literature, sometimes under other names. Basically, in the optimization framework, the regret acounts for the "loss" of choosing the point used in the algorithm over the best possible choice: the optimum. Therefore, we measure the difference between the point used/recommended by the algorithm and the optimum in terms of the objective function.

In general, an optimization algorithm searches for the optimum, and to do so, it produces iteratively *search points* which will be evaluated through the objective function. And at regular steps, the algorithm must return a *recommendation point* that will be the best approximation to the optimum so far.

Note that the recommendation point can be equal to a search point, but not necessarily.

The most usual form of regret is termed *Simple Regret*. The *Simple Regret* measures the distance, in terms of fitness values, between the optimum and the recommendation point output by the algorithm. It is widely used (possibly without this name) in noisy optimization [18, 14, 12]. However, some test beds, notably the Bbob/Coco framework in the first version, did not allow the distinction between search points (at which the fitness function is evaluated) and recommendations (which are output by the algorithm as an approximation of the optimum), so that the Simple Regret can not be checked. This leads to the use of an *Approximate Simple Regret* (name by us), which evaluates the fitness difference between the search points (and not the recommendations) and the optimum. Later, another form of regret, that we will term here *Robust Simple Regret*, was also proposed, using recommendation points. We analyze in this paper the use of different regrets that aim to estimate the quality of the approximated optimum in a similar way. In particular, we show to which extent they lead to incompatible performance evaluations of the same algorithm over the same class of noisy optimization problems, i.e. the convergence rate for the Approximate or the Robust Simple Regret overestimates or underestimates the convergence rate of the more natural simple regret. We also prove some new results in terms of Simple Regret itself.

2 Framework and Regrets

This section is devoted to the formalization of the noisy optimization problem considered and the analysed regrets. We will focus on the *Simple Regret* and the alternative definitions that aim to approximate it (denoted here *Approximate Simple Regret* and *Robust Simple Regret*). At the end of the section we will highlight some general relationships between the presented regrets.

2.1 Continuous Noisy Optimization

Given a fitness function $F : D \subset \mathbb{R}^d \to \mathbb{R}$, also known as objective function, optimization (minimization) is the search for the optimum point x^* such that $\forall x \in D, F(x^*) \leq F(x)$. The fitness function is corrupted by additive noise. In other words, given a search point $x \in D$, evaluating F in x results in an altered fitness value $f(x, w)$ as follows:

$$f(x, w) = F(x) + w, \tag{1}$$

where w is an independent random variable of mean zero and variance σ. In the present paper, we will consider a simple case, namely a standard Gaussian additive noise [1]. In addition, we assume that $F(x) = \|x - x^*\|^2$, where x^* is randomly uniformly drawn in the domain D. *Noisy optimization* is then the search for the optimum x^* such that $\mathbb{E}_w f(x^*, w)$ is approximately minimum, where $\mathbb{E}_w$ denotes the expectation over the noise w.

Consider a noisy black-box optimization scenario: for a point x, the only available information is the noisy value of F in x as given by $f(x, w)$ for some

[1] More general cases such that $\mathbb{E}f(x, w) = F(x)$ can be considered, as most algorithms do not request the noise to be additive and independent; the key point is the absence of bias.

independent w. An optimization algorithm generates $x_1, x_2, \ldots, x_n, \ldots$, successive *search points* at which the objective function is evaluated in a noisy manner. It also generates $\tilde{x}_1, \tilde{x}_2, \ldots, \tilde{x}_n, \ldots$ which are *recommendations* or *approximations of the optimum* after n fitness evaluations are performed.

2.2 Simple Regret and variants

The *Simple Regret* (SR) focuses only on approximating the optimum in terms of fitness values. Its definition is:

$$SR_n = \mathbb{E}_w \left(f(\tilde{x}_n, w) - f(x^*, w) \right) = F(\tilde{x}_n) - F(x^*)$$

Notice that the expectation operates only on w in $f(\tilde{x}_n, w)$, and not on $\tilde{x}_n$. As a consequence SR_n is a random variable due to the stochasticity of the noisy evaluations of the search points or the (possible) internal randomization of the optimization algorithm.

The SR can be a part of the performance evaluation of an algorithm. In the noise-free case it can be used to determine the *precision* of a method, by ensuring that the algorithm outputs a recommendation $\tilde{x}_m$ satisfying $SR_m \leq \epsilon$. Even more, when testing algorithms, it is common to use the "first hitting time" (FHT). FHT in fact refers to the first "stable" hitting time, i.e. the next recommendation is at least as good as the previous one. This is a reasonable assumption for algorithms solving noise-free problems. In this case the FHT is the minimum n such that $SR_n \leq \epsilon$, provided that the recommendation is defined as $\tilde{x}_n = x_{i(n)}$ with $1 \leq i(n) \leq n$ minimizing $SR_{i(n)}$. However, there is no exact equivalence or natural extension for the FHT with precision ϵ on noisy optimization. The algorithm only has access to noisy evaluations hence it cannot compute with "certainty" SR_n, which corresponds to the precision of the algorithm.

An alternative definition, that aims to measure the precision in a similar way to SR, is the *Approximate Simple Regret*[2] (ASR), defined by:

$$ASR_n = \min_{m \leq n} F(x_m) - F(x^*).$$

ASR takes in account the "best" evaluations among all the search points. It is used in the Bbob/Coco framework [7, 8, 19, 9, 26, 27, 23, 31, 22], and in some theoretical papers [15]. Notice that since ASR is non-increasing, the FHT can be computed.

In this paper we will also discuss another variant of regret, the *Robust Simple Regret*[3] (RSR), defined by:

$$RSR_n = \min_{k \leq n} \max_{(k - \lfloor g(k) \rfloor) < m \leq k} \left(F(\tilde{x}_m) - F(x^*) \right),$$

where $g(n)$ is a polylogarithmic function of n and $\lfloor \cdot \rfloor$ is the floor function. The RSR is the "best" SR since the beginning of the run, sustained during $\lfloor g(k) \rfloor$ consecutive recommendations [4]. The polylogarithmic nature of $g(\cdot)$ is

[2] The name is proposed by us.

[3] Discussed on Bbob-discuss mailing list (`http://lists.lri.fr/pipermail/bbob-discuss/2014-October.txt`. The name is proposed by us.

[4] If $(k - \lfloor g(k) \rfloor) < 0$, then the max on the definition considers indexes between 1 and k

explained by the following argument: $g(k)$ be large enough, so that we have a correct recommendation confirmed over $g(k)$ iterations, but small enough, so that we do not have to wait many evaluations before acknowledging that a correct recommendation has been found. The RSR uses the recommendation $\tilde{x}_m$ instead of the search point x_m used in ASR. But it uses the worst of a sequence of recommendations.

As a side note about the definition of RSR, it was originally proposed to use a quantile instead of the maximum. The "quantile version" (without this name), was proposed to become part of the performance measure in Bbob/Coco. However, we will show that it is possible to get a RSR decreasing quicker than the SR, so that RSR is a poor approximation of SR. The result is valid even with the quantile 100%, i.e. the maximum. The same is possible with any other quantile.

The introduction of RSR apparently outplays ASR as an approximation of SR by two means. First, by using recommendations rather than search points. Second, by checking on multiple recommendations that the optimum is correctly found with a given precision. In addition, as well as ASR, it is non-increasing, therefore it can be used for fastening experiments on testbed. Please note however that this advantage makes sense only when the target fitness value is known, which is rarely the case except in an artificial testbed.

To investigate the convergence rate of the regrets, we will use a slightly different notation than classical works on noisy optimization. Usually the rates are given in terms of $O(h(n))$ where $h(n)$ is some function depending on the number of evaluations n. The state of the art shows that in many cases ([17, 18, 29, 15]) $SR_n = O(n^{\psi})$ where $\psi < 0$ implies that the algorithm converges. Therefore, there is a linear relationship between $\log(SR_n)$ and $\log(n)$ with a *slope* ψ, where $\log(\cdot)$ is the natural logarithm. We will then refer to the *slope of the regret* when speaking about the convergence rate of the regret. The definition of the *slope of the SR* is:

$$s(SR) = \limsup_{n\to\infty} \frac{\log(SR_n)}{\log(n)}$$

We have the corresponding definition for the slope of ASR and RSR. Notice that if the slope is close to 0, then the algorithm (at best) converges *slowly*. On the contrary, if the slope is negative and far away from 0, then the algorithm is *fast*.

2.3 General results for SR, ASR and RSR: RSR is an optimistic approximation of SR

The problems analysed in this paper arise from the gap between $s(SR)$ and $s(ASR)$ or $s(RSR)$. Ideally we would like to have a regret that can be used easily and that *approximates* the Simple Regret. In the following sections (3 and 4) we will see with specific examples there is indeed a gap between $s(SR)$ and $s(ASR)$. In some cases using ASR overestimates the performance of algorithms and in others it underestimates their performance. An extreme case is detailed in section 4, where we prove that Alg. 3 has optimal convergence rate in term of SR whilst for ASR it does not converge at all.

In general, by definition, we have that for any algorithm, $s(RSR) \leq s(SR)$. From this point of view, RSR is a correct lower bound for SR. In other words,

if an algorithm is fast in terms of SR, its performance measured by RSR will be at least as good. Unfortunately, this bound is not nearly tight. We will prove that a small modification on the algorithm induces $s(RSR) \leq s(ASR)$ whereas $s(SR)$ is the same - so that, for cases in which $s(ASR) < s(SR)$ (sometimes by far, as explained in later sections), we get $s(RSR) < s(SR)$ (sometimes by far).

Let A be an optimization algorithm and its search points $(x_i)_{i\geq 1}$. Consider another algorithm denoted A_g and its successive search points $(X_i)_{i\geq 1}$. The search points of A_g are obtained by repeating $\lfloor g(n) \rfloor$ times, for any n, the search points x_n of A. Hence, we get the assignment:

$$\begin{array}{rcccl} X_1 = & \dots & = & X_{1+\lfloor g(1)\rfloor} & \leftarrow x_1 \\ X_{2+\lfloor g(1)\rfloor} = & \dots & = & X_{2+\lfloor g(1)\rfloor+\lfloor g(2)\rfloor} & \leftarrow x_2 \\ & \vdots & & & \\ X_{n+\sum_{j=1}^{n-1}\lfloor g(j)\rfloor} = & \dots & = & X_{n+\sum_{j=1}^{n}\lfloor g(j)\rfloor} & \leftarrow x_n \end{array}$$

Let the recommendation points of A_g be defined by $\tilde{X}_n = X_n$ for any n. Therefore A_g is a slightly modified version of A since there is an additional polylogarithmic number of evaluation in A_g, assuming g is polylogarithmic. The RSR of A_g (say RSR_{A_g}) converges approximately as fast as the ASR of the original algorithm (say ASR_A). The extra polylogarithmic number of evaluations does not affect the linear convergence in log/log scale. Hence, for any algorithm A, $s(RSR_{A_g}) \leq s(ASR_A)$.

The general relationships between the slopes of the SR and its approximations are not conclusive since the bounds are not tight. We will show some gaps between the different approximations of SR and the SR itself. In the following sections we present five algorithms that will serve as clear examples to see the differences of using one or another regret as performance measures. We will focus in two types of algorithms: the first group, in Section 3, consists of Evolutionary Algorithms and Random Search and the second, in Section 4, of algorithms using approximations of the gradient of the objective function. For each class of algorithms, we exhibit convergence rate bounds on $s(SR)$, $s(ASR)$ and $s(RSR)$. Section 5 displays some experimental works in order to confront theory, conjecture and practice.

3 Evolutionary Algorithms

On the group of Evolutionary Algorithms (EAs), we present Random Search (RS), Evolution Strategy (ES) and Evolution Strategy with resampling (ES+r). They all use comparisons between fitness values to optimize the function.

3.1 Random Search

Random Search (Alg. 1) is the most basic of stochastic algorithms [24]. The search points $x_1, \ldots, x_n, \ldots$ are independently identically drawn according to some probability distribution. $\tilde{x}_n$ is usually the search point with the best fitness so far, i.e. with y_i the fitness value obtained for x_i, we have $\tilde{x}_n = x_i$ with $i \in \{1, \ldots, n\}$ minimizing y_i.

Algorithm 1 Random Search.

```
Initialize: Candidate solution x̃ randomly drawn in [0,1]^d
bestfitness ← f(x̃)
Initialize: n ← 1
while not terminated do
    Randomly draw y in [0,1]^d.
    fitness ← f(y)
    if fitness < bestfitness then
        x̃ ← y
        bestfitness ← fitness
    end if
    n ← n + 1
end while
    return x̃
```

We consider in this paper a simple variant of RS to show clearly the contrast between SR and ASR.

Framework for RS: each search point is randomly drawn independently and uniformly, sampled once and only once, with the uniform probability distribution over $[0,1]^d$. The objective function is the noisy sphere function f:

$$f(x) = \|x - x^*\|^2 + \mathcal{N}. \tag{2}$$

where $\mathcal{N}$ is a standard Gaussian variable.

In this setting, existing results in the literature imply a bound on $s(ASR)$ as explained in Property 1. We will prove then that the slope of the Simple Regret is not negative, as formalized in Theorem 1.

3.1.1 Approximate Simple Regret: $s(ASR) = -2/d$

Property 1. *Consider RS described in Alg. 1, with the framework above. Then almost surely* $ASR_n = O\left(\frac{1}{n^{2/d}}\right)$.

Proof. [16] has shown that among n points generated independently and uniformly over $[0,1]^d$ the closest search point to the optimum is almost surely at distance $O\left(\frac{1}{n^{1/d}}\right)$ from the optimal point x^* within a logarithmic factor. Hence for the sphere function[5], the Approximate Simple Regret ASR_n almost surely satisfies: $ASR_n = O\left(\frac{1}{n^{2/d}}\right)$ up to logarithmic factors. □

3.1.2 Simple Regret: $s(SR)$ is not negative

Theorem 1. *With the framework above, for all* $\beta > 0$, *the expected simple regret* $\mathbb{E}(SR_n)$ *is not* $O(n^{-\beta})$.

Proof. See supplementary material. □

[5] The result also holds for a function locally quadratic around a unique global optimum.

Remark: Roughly speaking, the proof of the theorem is based on the fact that with a non-zero probability a search point which does not have the best fitness value, is selected as the best point in Lines 7-9 of Alg. 1.

3.2 Evolution Strategies (ES)

Evolution Strategies [25, 28] are algorithms included in the category of Evolutionary Algorithms (EAs). In general, EAs evolve a population until they find an optimum for the objective or fitness function. The process starts by a population randomly generated. Then the algorithm iterates creating new individuals using crossover and mutation and then evaluating this new population of offspring and selecting the ones - regarding to their fitness values - that will become the parents of the next generation.

ES have some more specific selection and mutation processes. Usually the mutation is performed by creating new individuals starting from the parent and adding a random value to it (usually normally distributed around the parent). There are various rules for choosing the step-size.The selection in ES is usually deterministic and rank-based. This is, the individuals chosen to be the parents of the next generations are the ones that have the best fitness values.

When dealing with noisy function, the sorting step of the ES is disturbed by the noise and misranking might occur. To tackle this problem, Arnold and Beyer, in [1, 2] propose to increase the population size. An alternative is to evaluate multiple times the same search point and average the resulting fitness values. For a given search point $x \in D$, r evaluations are performed: $(f(x, w_1), \ldots, f(x, w_r))$ and the fitness value used in the comparisons is the average of these evaluations $\frac{1}{r}\sum_{i=1}^{r} f(x, w_i)$. In particular, the variance of the noise is divided by r. Several rules have been studied: constant [13], adaptive (polynomial in the inverse of the step-size), polynomial and exponential [5] number of resamplings. A general (μ, λ)-ES is presented in Algorithm 2.

Algorithm 2 (μ, λ)-Evolution Strategy. The resampling function r may be constant or depend on the number of iterations and possibly on the step-size. When $r = 1$, the algorithm reduces to an ES without resampling. $\mathcal{N}$ is a standard Gaussian of dimension d. Here, the index n is the number of *iterations*

Input: Parameters μ, λ and resampling function r
Initialize: Parent $\tilde{x}$ and Step-size σ
Initialize: $n \leftarrow 1$
while not terminated **do**
  Mutation step: $\forall i \in \{1, \ldots, \lambda\}$, $x^{(i)} \leftarrow \tilde{x} + \sigma\mathcal{N}$
  Evaluation step:
    $\forall i \in \{1, \ldots, \lambda\}$ $y^{(i)} \leftarrow \frac{1}{r(n)}\sum_{j=1}^{r(n)} f(x^{(i)}, w_j)$
  Selection step: Sort the population according to their fitness and select the μ best: $(x^{(i)})^{\mu}$
  Update Parent: $\tilde{x}$ from σ, $(y^{(i)})^{\mu}$ and $(x^{(i)})^{\mu}$
  Update Step-size: σ from σ, $(y^{(i)})^{\mu}$ and $(x^{(i)})^{\mu}$
  $n \leftarrow n + 1$
end while
  return $\tilde{x}$

3.2.1 Regrets for ES without resampling

It is known [3] that when the noise strength is too big, classical evolution strategies (without reevaluations or other noise adaptation scheme) do not converge, they stagnate. [10] experimentally shows that an ES without any adaptation to

the noisy setting stagnates around some step-size and at some distance of the optimum. The divergence results suggest that the ES in this case is only as a more sophisticated version of RS. The steps of the ES are more complicated, but not sufficiently adapted to handle the noise of the function. We propose then a Conjecture on the convergence rates for ES.

Conjecture 1 (Convergence rates for ES)**.** *Evolution Strategies without a noise handling procedure have the same convergence rates as Random Search for all regrets.*

3.2.2 Simple regret for ES with resamplings

We will see that the results are more encouraging than in Section 3.2.1 when we consider an ES with some adaptation to mitigate the effect of the noise. We will assume that the function r (number of revaluations per point) in Alg. 2 grows polynomially or exponentially with the number of iterations.

The work in [5] shows that ES that include an exponential number of revaluations converges with high probability to the optimum. The convergence rate is $s(SR) = K$ for some $K < 0$ under assumptions about the convergence in ES in the noise-free case. Moreover [4] shows that ES, under general conditions, must exhibit $K > -\frac{1}{2}$. There is no formal proof of an upper bound that can theoretically ensure a value or a range for $s(SR)$. However, the experiments on [5] suggest that the $K = -\frac{1}{2}$ is reached for functions with a quadratic Taylor expansion and additive noise (as in Eq. 1). Hence we propose Conjecture 2:

Conjecture 2 (SR for ES + r)**.** *Consider* $0 < \delta < 1$. *For some resampling parameters (i.e. for some revaluation function* r*), Evolution Strategies with resampling (Alg. 2) satisfy* $s(SR) = -1/2$ *with probability* $1 - \delta$.

This conjecture applies to some ES with step-size scaling as the distance to the optimum, i.e. σ_n used for generating the nth search point has the same magnitude as $\|\tilde{x}_n - x^*\|$ ([25, 11]). [10] has proposed variants of ES for quickening the convergence thanks to large mutations and small inheritance. Such an approach is not covered by the bound in [4] and it is for sure an interesting research direction - maybe it might reach slope $s(SR) = -1$.

3.2.3 Approximate simple regret for ES with resamplings

We have seen that an ES can reach a slope of SR approximately $-\frac{1}{2}$, when using resamplings. However, ASR can be better by slightly modifying the original ES, and therefore achieving a faster convergence rate than the real one represented by $s(SR)$ We consider an ES - called $MES{+}R$ for Modified ES with Resampling. Let r_n be exponential in the number of iterations: $r_n = R \cdot \zeta^n$, $R > 0$, $\zeta > 1$. $MES + R$ is as in Alg. 2 with the following modifications. At iteration n:

Generation: (Alg. 2, Line 5) Generate additional r_n "fake" offspring $\{x^{(i)f} : 1 \leq i \leq r_n\}$, with the same probability distribution as the λ offspring. They will be evaluated one time each, but they will *not* be taken into account for the selection. Note that this means that they are part of the sequence of points considered by ASR, but not by SR.

Evaluation: (Alg. 2, Line 6) Evaluate r_n times each "true" offspring $\{x^{(i)} : 1 \leq i \leq \lambda\}$ to obtain their corresponding fitness value $y^{(i)}$. Evaluate one time

each "fake" offspring. Therefore, performing $(\lambda+1)r_n$ function evaluations in each iteration.

Then, under some reasonable convergence assumptions which are detailed in theorem 2 below, the ASR reaches a faster rate: $s(ASR) = -1/2 - 2/d$ with high probability.

Theorem 2. *Consider $0 < \delta < 1$. Consider an objective function $F(x) = \|x\|^2$, where $x \in \mathbb{R}^d$. Consider a $MES + R$ as described previously. Assume that:*

(i) σ_n and $\|\tilde{x}_n\|$ have the same order of magnitude:

$$\|\tilde{x}_n\| = \Theta(\sigma_n). \tag{3}$$

(ii) $\log - \log$ *convergence occurs for the SR:*

$$\frac{\log(\|\tilde{x}_n\|)}{\log(n)} \underset{n\to+\infty}{\longrightarrow} -\frac{1}{2} \quad \textit{with probability } 1-\delta, \tag{4}$$

Then, with probability at least $1-\delta$, $s(ASR) = -1/2 - 2/d$.

Proof. See supplementary material. □

Remark 1. *The assumption of $s(SR) = -1/2$ is based on the convergence of ES in the noise-free case and it is essential to prove Theorem 2. This rate of convergence can be proved when the ES converges in the noise-free case (details on [5]). But the convergence of ES has not been formally proved, not even for the noise-free case. There is an important element given in [6], showing that $\frac{1}{n}\log||x_n - x^*||$ converges to some constant, but this constant is not proved negative. Furthermore, parameters ensuring convergence in the noisy case are unspecified in [5].*

4 Stochastic Gradient Descent

For the group of Stochastic Gradient Descent Algorithms, we consider the ones presented by Shamir [29] and Fabian [18] which approximate the gradient of the objective function. We will denote them Shamir algorithm and Fabian algorithm respectively. Both of them approximate the gradient of the function using function evaluations by different methods, therefore they remain in the black-box category. Fabian algorithm uses the average of redundant finite differences and Shamir algorithm a one point estimate gradient technique.

The convergence rates in terms of SR are proved in [29] and [18]. For Shamir it is shown that $s(SR) = -1$ in expectation for quadratic functions. Fabian ensures a rate $s(SR) = -1$ approximately and asymptotically only for limit values of parameters. However, it requires only smooth enough functions, so the class of functions is wider than the one considered in [29]. This rate $s(SR) = -1$ has been proved tight in [14]. Hence, Shamir and Fabian algorithm are faster than ES's, which cannot do better than $s(SR) = -\frac{1}{2}$, at least under their usual form [4].

Algorithm 3 Shamir Algorithm for Quadratic functions. Π_W represents the projection over the space W

Input: Parameters λ and ϵ
Initialization: $\hat{x}_1 \leftarrow 0$, $n \leftarrow 1$
while not terminated **do**
  Pick $r \in \{-1, 1\}^d$ uniformly at random
  $x_n \leftarrow \hat{x}_n + \frac{\epsilon}{\sqrt{d}} r$
  Evaluate: $v \leftarrow f(x_n, w)$
  $\hat{g} \leftarrow \frac{\sqrt{d} v}{\epsilon} r$
  $\hat{x}_{n+1} \leftarrow \Pi_W \left(\hat{x}_n + \frac{1}{\lambda n} \hat{g}\right)$
  Recommend: $\tilde{x}_n \leftarrow \lceil \frac{2}{n} \rceil \sum_{k=\lceil n/2 \rceil}^{n} \hat{x}_k$
  $n \leftarrow n + 1$
end while
  return $\tilde{x}_n$

4.1 Shamir's quadratic algorithm

Shamir algorithm presented in [29] for quadratic functions is Algorithm 3.

One of the key points in Alg. 3 are that there is only one evaluation per iteration (somehow in the spirit of Simultaneous Perturbation Stochastic Approximation SPSA [30, 20]). The second important point is that the expectation of the distance between search points and recommendations is constant, which implies that the search points do not converge towards the optimum! This is not a problem for the convergence in terms of SR, when search points x_n and recommendations $\tilde{x}_n$ are distinguished, but it makes a difference for ASR.

Shamir algorithm has an optimal convergence rate in expectation ($s(SR) = -1$) for quadratic functions. This fact should be acknowledge by any other regret used to evaluate its performance which aims to aproximate the SR. But intuitively in the framework of Shamir algorithm, the $s(ASR)$ is presumably a bad approximation of $s(SR)$ due to the queries at a constant distance of the current recommendation. This convergence rate in terms of $s(ASR)$ could not be obtained from the results in [29]. Nonetheless, we prove in a general way that as long as the results for Shamir algorithm are satisfied *almost surely*, then $s(ASR) = 0$ a.s. Therefore we present the latter result in Theorem 3 and a conjecture on the convergence rate of $s(ASR)$ in expectation for Shamir algorithm.

Theorem 3. *Assume that the optimum x^* is unique and that $(\tilde{x}_n)$ is a sequence of recommendation points converging a.s. to x^*. Assume that the sequence of evaluation points (x_n) is such that $\forall n, x_n \neq x^*$ a.s. and that $\|x_n - \tilde{x}_n\|$ is constant. Then, a.s.*

$$s(ASR) = 0.$$

Proof. $\tilde{x}_n$ converges almost surely to the optimum and x_n is at a constant distance from $\tilde{x}_n$. Therefore the distance between x_n and the optimum converges to a constant. This implies that the minimum $\min_{i=1}^{n} \|x_i - x^*\|^2$ is lower bounded by some constant. Therefore $s(ASR) = 0$. □

Conjecture 3 (ASR for the Shamir algorithm)**.** *Shamir algorithm also verifies $s(ASR) = 0$ a.s. on quadratic functions.*

4.2 Fabian Algorithm

Algorithm 4 presents the algorithm studied in [18]. Unlike Algorithm 3, Fabian algorithm performs several evaluations per iteration, and the distance between search point and recommandation decreases.

Algorithm 4 Fabian Algorithm. e_i is the ith vector of the standard orthogonal basis of $\mathbb{R}^d$ and $e_{1,s/2}$ is the 1st vector of the standard orthogonal basis of $\mathbb{R}^{s/2}$. v_i is the ith coordinate of vector v. $(\hat{x}_i)$ is the ith coordinate of intermediate points $(\hat{x})$. $(x^{(i,j)+})$ and $(x^{(i,j)-})$ are the search points and $\tilde{x}$ is the recommendation. Here, the index n is the number of *iterations*.

Input: An even integer $s > 0$. Parameters a, α, c, γ.
Initialization:
$u_i \leftarrow \frac{1}{i}$, $\forall\ i \in \{1, \dots, s/2\}$
Matrix $U \leftarrow \left(\|u_j^{2i-1}\|\right)_{1 \le i,j \le s/2}$
Vector $v \leftarrow \frac{1}{2} U^{-1} e_{1,s/2}$
$\tilde{x} \leftarrow x \in [0,1]^d$ uniformly at random
$n \leftarrow 1$
while not terminated **do**
  $a_n \leftarrow \frac{a}{n^\alpha}$, $c_n \leftarrow \frac{c}{n^\gamma}$
  $\forall j \in \{1, \dots, s/2\}$, $\forall i \in \{1, \dots, d\}$
  Evaluate:

$$x^{(i,j)+} \leftarrow \tilde{x} + c_n u_j e_i \qquad x^{(i,j)-} \leftarrow \tilde{x} - c_n u_j e_i$$

  $\hat{x}_i \leftarrow \frac{1}{c_n} \sum_{j=1}^{s/2} v_j \left(f(x^{(i,j)+}) - f(x^{(i,j)-})\right)$
  Recommend: $\tilde{x} \leftarrow \tilde{x} - a_n \hat{x}$
  $n \leftarrow n + 1$
end while
  return $\tilde{x}$

The work in [18] gives the convergence rate in terms of SR. The result is presented here as Theorem 4. The value of the $s(SR)$ depends on the parameters of the algorithm and it is ensured a.s.

Theorem 4 (Simple Regret of Fabian algorithm)**.** *Let s be an even positive integer and F be a function $(s+1)$-times differentiable in the neighborhood of its optimum x^*. Assume that its Hessian and its $(s+1)^{th}$ derivative are bounded in norm. Assume that the parameters given in input of Algorithm 4 satisfy: $a > 0$, $c > 0$, $\alpha = 1$, $0 < \gamma < 1/2$ and $2\lambda_0 a > \beta_0$ where λ_0 is the smallest eigenvalue of the Hessian. Let $\beta_0 = \min(2s\gamma, 1 - 2\gamma)$. Then, a.s.:*

$$n^\beta (\tilde{x}_n - x^*) \to 0 \ \forall\ \beta < \beta_0/2 \tag{5}$$

In particular, when F is smooth enough, we get $s(SR) = -2\beta$.

Remark 2. *Note that $s(SR)$ optimal when $\gamma = \frac{1}{2}(s+1)^{-1}$. In this case, $\beta_0 = \frac{s}{s+1} \underset{s \to \infty}{\to} 1$. β_0 can be made arbitrarily close to 1, so 2β also, but then γ goes to 0. Hence we get the values of Table 1, with $2\beta = 1 - e$, $e > 0$ and close to 0.*

This shows that the Fabian algorithm can have $s(SR)$ arbitrarily close to -1, which is optimal. As in the case of Shamir, this optimal performance should be captured by the regret used to evaluate the algorithm. Unfortunately, this is not the case, as detailed in Theorem 5.

Theorem 5 (ASR of Fabian algorithm)**.** *Let F be a λ-convex and μ-smooth function corrupted by an additive noise with upper bounded density and with optimum randomly drawn according to a distribution with upper bounded density. Then, a.s.,*

$$s(ASR) = -\min(2\beta, 2\gamma).$$

Proof. See supplementary material. □

Remark 3. *Theorem 5 shows that $s(ASR) = -\min(2\beta, 2\gamma)$, i.e., when then Fabian algorithm is optimized for SR, $s(ASR)$ is close to -2γ, close to 0.*

4.3 Shamir and Fabian adapted for ASR

Both algorithms presented in this section have a clear difference between the search and recommendation points. This fact is not automatically distinguished when we are evaluating their performance using for example a test bed. If we modify the algorithms we can achieve ASR approximating well the optimal behavior reported by SR. A modification such as sampling one point out of two at the current recommendation, without using it in the algorithm can imply $s(ASR) = s(SR)$ arbitrarily close to -1.

	SR		ASR		RSR	
	conv.	type	conv.	type	conv.	type
Evolutionary Algorithms						
RS	$\mathbf{0}$	**expect.**	$\mathbf{-\frac{2}{d}}$	**a.s.**	$\mathbf{-\frac{2}{d}}$	a.s.
ES	0	expect.	$-\frac{2}{d}$	a.s.	$-\frac{2}{d}$	a.s.
ES + r	$-\frac{1}{2}$	high prob.	$-\frac{1}{2}$	high prob.	$-\frac{1}{2}$	high prob.
MES+ r	$-\frac{1}{2}$	high prob.	$\mathbf{-\frac{1}{2}-\frac{2}{d}}$	**high prob.**	$\mathbf{-\frac{1}{2}-\frac{2}{d}}$	**high prob.**
Stochastic Gradient						
Shamir	$\mathbf{-1}$	**expect.**	$\mathbf{0}$	**expect.**	$\mathbf{-1}$	**expect.**
Shamir for ASR	$\mathbf{-1}$	**expect.**	$\mathbf{-1}$	**expect.**	$\mathbf{-1}$	**expect.**
Fabian	$\mathbf{-(1-e)}$	**a.s.**	$\mathbf{-e'}$	**a.s.**	$\mathbf{-(1-e)}$	**a.s.**
Fabian for ASR	$\mathbf{-(1-e)}$	**a.s.**	$\mathbf{-(1-e)}$	**a.s.**	$\mathbf{-(1-e)}$	**a.s.**

Table 1: Convergence rates for the regrets analysed on this paper. The "convergence" column refers to the convergence rate and the "type" column specifies the type of convergence: with high probability, in expectation, almost surely. The results in bold are proved and the others are conjectures, all of them presented in this paper.