Lower Bounds for Evolution Strategies

Olivier Teytaud

Publishers' page

Publishers' page

Publishers' page

Publishers' page

Contents

Chapter 1

Lower Bounds for Evolution Strategies

Olivier Teytaud
TAO (Inria), Lri, Umr 8623 (Cnrs, Univ. Paris-Sud),
Bât. 490, Univ. Paris-Sud, 941405 Orsay, France, teytaud@lri.fr

The mathematical analysis of optimization algorithms involves upper and lower bounds; we here focus on the second case. Whereas other chapters will consider black box complexity, we will here consider complexity based on the key assumption that the only information available on the fitness values is the rank of individuals - we will not make use of the exact fitness values. Such a reduced information is known efficient in terms of robustness (Gelly *et al.*, 2007), what gives a solid theoretical foundation to the robustness of evolution strategies, which is often argued without mathematical rigor - and we here show the implications of this reduced information on convergence rates. In particular, our bounds are proved without infinite dimension assumption, and they have been used since that time for designing algorithms with better performance in the parallel setting.

1.1 INTRODUCTION

Optimization methods in continuous domains (we will consider also discrete cases in this chapter) can be classified in a few categories:

- order 2 methods, computing fitness values, gradients and Hessians;
- order 1 methods, computing fitness values and gradients only;
- order 0 methods, computing fitness values only.

However, the order of the optimization algorithm sometimes refers to the convergence rate of the algorithm for some family of fitness functions: order

$q > 1$ means that, with y_k the fitness value of the best evaluated point after k steps,

$$(y_{k+1} - y^*)/(y_k - y^*)^q = O(1).$$

Order $q = 1$ means that

$$(y_{k+1} - y^*)/(y_k - y^*) \leqslant c \text{ for some } c < 1.$$

These two definitions do not coincide, for several reasons:

- Newton methods are of order 2 for both definitions (at least mathematically, as limited precision often reduce the "real world" order of convergence).
- Quasi-Newton methods have a superlinear convergence; therefore, they are "a bit more" than order 1 in spite of the fact that they use only gradients (they approximate the Hessian thanks to successive gradients, but they never compute it explicitly).
- Algorithms with no gradient can nonetheless be linear. This is the case for most evolution strategies (Auger, 2005). Also, thanks to approximation of the objective function built on previously visited points, algorithms with no gradient can be superlinear; (Powell, 1974), cited by [Conn *et al.* (1997)], shows that, in dimension 1, the secant model has average order $1.618\cdots = \frac{1}{2}(1+\sqrt{5})$, i.e. more than the Newton-Raphson algorithm with finite differences which reaches $\sqrt{2}$.

Evolution strategies (ES) are often termed "order 0" methods as they do not use gradients or Hessians. Interestingly, most ES are in fact a special case of order 0 methods: in addition to not using gradients, they only use comparisons between fitness values and not the fitness values themselves - *i.e.* they have only access to a discrete information, with finitely many possible values. Using comparisons only makes an important difference since it is known that, even without gradient, a super-linear convergence can be obtained when using fitness values (as done by (Auger *et al.*, 2005), using surrogate models for a super-linear convergence rate), whereas it is not possible with comparison-based non-elitist algorithms. Getting rid of the gradient or Hessian is easy to justify: often, the gradient is quite expensive, and the Hessian is even much more expensive - sometimes, they both just do not exist. But is there a good reason for considering algorithms using only comparisons and discarding the other information provided by the fitness values?

A first answer lies in the optimality of this comparison-based principle for some robustness criterion; this robustness was suggested by [Bäck *et al.* (1991); Whitley (1989); Baker (1987)] and was formally established by [Gelly *et al.* (2007)]. The robustness criterion is the supremum over compositions with monotonic functions: instead of

$$\underbrace{\mathbb{E}_{f\in F}}_{\text{sup. on fitness}} \quad \mathbb{E}\|x_{N,f} - x^*(f)\|^2$$

where $x_{N,f}$ is the approximation of the optimum of f provided by the Alg. after N visited points and $x^*(f)$ is the optimum of f, the robust criterion considered by (Gelly *et al.*, 2007) is

$$\underbrace{\mathbb{E}_{f\in F}}_{\text{sup. on fitness}} \sup_{g \text{ monotonic}} \mathbb{E}\|x_{N,g\circ f} - x^*(f)\|^2$$

and for this criterion, there is an optimal algorithm which is comparison-based. A second answer is natural in the case of problems in which you have only comparisons between parametrizations, but no fitness values: for example, when tuning a strategy in transitive games, or in interactive optimization when the comparison is provided by a human.

In many cases, comparisons between individuals are used in order to select a subset of the population; these selected individuals are the parent of the next generation. However, they can also be used for ranking these selected individuals or even the whole population. Examples of algorithms using more information than just the selection of a subset are the roulette-wheel with rank-based fitness assignment (stochastic sampling (Baker, 1987), rank-based fitness assignment (Bäck *et al.*, 1991; Whitley, 1989)), weighted recombination (Hansen and Ostermeier, 2001; Arnold, 2005) or BREDA (Gelly *et al.*, 2007).

Hence, the wide family of order 0 methods can be divided, from the more general to the more specific case, into (i) algorithms using fitness values; (ii) algorithms using the full ranking of all the population; (iii) algorithms using the full ranking of selected points; (iv) algorithms using only the selection information (this will be formalized in Section 1.2). In cases (ii) to (iv), the branching factor of the algorithm, i.e., the number of possible outcomes for the information extracted from the population in one iteration, is finite and bounded. This simple fact has been used by (Teytaud and Gelly, 2006) in order to provide lower bounds that match some upper bounds known for evolutionary algorithms (Droste, 2005; Auger, 2005; Rudolph, 1997);

this simple technique extends previous lower bounds shown by (Witt and Jägersküpper, 2005).

We will first introduce formal notations in Section 1.2 and background in Section 1.3, so that we can formalize the main results. Lower bounds on $(\mu \dot{+} \lambda)$-ES[1] based on the branching factor, obtained by [Teytaud and Gelly (2006)], are then recalled (Section 1.4). However, improvements under some mild assumptions were proved by (Fournier and Teytaud, 2008); these results, based on VC-dimension, are presented in Section 1.6. The traditional case of the sphere function is studied in Section 1.7. We also present a simple algorithm based on full ranking, which allows to obtain an almost linear speed-up[2] when the size of the offspring is linear in the dimension. Conclusions and elements for further research are presented in Section 1.8.

Notations. In all the chapter, $\log(x)$ denotes the logarithm with basis 2, i.e. $\log(2) = 1$. The set of integers $\{1, 2, \ldots, n\}$ is denoted by $[[1, n]]$. The notation $|\cdot|$ is used to denote both the cardinal of a set and the length of a vector.

1.2 EVOLUTION STRATEGIES OF TYPE $(\mu \dot{+} \lambda)$

This section is devoted to a formal definition of evolution strategies of type $(\mu \dot{+} \lambda)$ (denotes $(\mu \dot{+} \lambda)$-ES). The aim of $(\mu \dot{+} \lambda)$-ES is to find the minimum of a real-valued function on a domain D ($f : D \to \mathbb{R}$); f is termed the fitness function. We will here focus on the classical case of $(\mu \dot{+} \lambda)$-ES which are not allowed to have access to the fitness value, but only to comparisons between fitness values: given x and y in D, the algorithm can request the sign of $f(x) - f(y)$. More precisely, given some points $z_1, \ldots, z_p \in D$ (usually termed population), these algorithms are given (possibly partial) information on the signs of $f(z_i) - f(z_j)$, for all $1 \leqslant i < j \leqslant p$: all the signs in the case of complete ranking of the population, only a subset of these signs when the algorithms only uses a selection information (these notions will be formalized below). Of course these algorithms are not required to work for only one fitness function, but for a whole family of fitness functions. In the following, we denote by $\mathcal{F}$ the set of fitness functions considered, and we will consider worst cases on functions in $\mathcal{F}$.

[1] $(\mu \dot{+} \lambda)$-ES stands for either $(\mu + \lambda)$-ES or (μ, λ)-ES.

[2] The speed-up is the ratio between the new convergence ratio and the old convergence ratio when a modification is performed; here, we compare the convergence ratio of the algorithm with λ linear in the dimension and the convergence ratio with λ constant.

In the rest of the chapter, otherwise explicitly stated, we assume that equality of fitness values $f(x) = f(y)$, for two points x and y generated at any iteration, never occurs. This is a reasonable hypothesis in the continuous setting (e.g., when $D = [0,1]^d$) where this assumption almost surely holds for many algorithms and many fitness functions; the case with equality is developped by (Fournier and Teytaud).

Let λ and μ be two integers. The first case is the case of Selection Based evolution strategies (SB-$(\mu \dot{+} \lambda)$-ES). In this case, there is a set $\mathcal{I}$ of internal states. The algorithm starts in the initial state I_0 returned by the function *initial_state*. At each iteration, the algorithm follows these three successive steps. First generate a set of λ points, called the *offspring*. Then select only the μ best ones, i.e. the μ points with lowest fitness values; in the case of an SB-(μ, λ)-ES (termed non-elitist), points generated at previous stages are forgotten and this selection is performed only among the offspring, while an algorithm of type SB-$(\mu + \lambda)$-ES (termed elitist) selects the μ best points among the offspring *and* the points selected at the previous step[3]. Finally, the internal state is updated: this might include for example cumulative step-length adaptation (CSA) by (Hansen and Ostermeier, 2001), covariance matrix adaptation (CMA) by [Hansen and Ostermeier (2001)], mutative self-adaptation (SA) by [Rechenberg (1973); Schwefel (1974)], covariance matrix adaptation by SA (CMSA) by [Beyer and Sendhoff (2008)], the one-fifth rule for step-size adaptation (Rechenberg, 1973; Beyer, 2001).

Classical ES could be cast into our framework; in the continuous case, just use Gaussian mutations with parameters encoded in the internal state. Please notice that the framework is in fact much more general than that: it includes adaptation by cross-entropy methods (CEM by [de Boer *et al.* (2005)]) or Estimation of Distribution Algorithms like e.g. UMDA (Mühlenbein and Mahnig, 2002), Compact Genetic Algorithm (Harik *et al.*, 1999), Population-Based Incremental Learning (Baluja, 1994), Relative Entropy (Mühlenbein and Höns, 2005) and Estimation of Multivariate Normal Algorithms (EMNA by [Larranaga and Lozano (2001)]); this similarity points out the strong similarity between evolution strategies, cross-entropy methods, estimation of distribution algorithms which are all in the scope of this paper. We can even include so-called direct search methods (Conn *et al.*, 1997), including the Nelder-Mead algorithm (Nelder and Mead, 1965), the Hooke&Jeeves algorithm (Hooke and Jeeves, 1961).

[3]As a consequence, in elitist cases, these μ selected points are always the μ points with lowest fitness values found so far.

Notice that $\mu \leqslant \lambda$ must hold in the case of an SB-(μ, λ)-ES.

Algorithm 1.1 (Fournier and Teytaud, 2008) Selection Based (μ, λ)-ES (resp. Selection Based $(\mu + \lambda)$-ES). Framework for evolution strategies based on selection, working on a fitness function f. The random variable ω is a random seed. An algorithm matching this framework is obtained by specifying the distribution of ω, the space of states, and the functions *initial_state*, *generate*, *update* and *proposal*.

Initialization: $I_0 \leftarrow \text{initial\_state}(\omega)$; $S_0 \leftarrow \emptyset$; $n \leftarrow 0$
while true **do**
  $n \leftarrow n + 1$
  Generation step (generate an offspring O_n of λ distinct points)**:** $O_n \leftarrow \text{generate}(I_{n-1})$
  $E_n \leftarrow O_n$ (resp. $E_n \leftarrow O_n \oplus S_{n-1}$)
  $\ell \leftarrow \min(\mu, |E_n|)$
  Selection step: $v_n \leftarrow \text{select}(E_n, f)$, i.e.:
    The vector $v_n = (i_1, \ldots, i_\ell)$ is defined by:

$$\begin{cases} 1 \leqslant i_1 < i_2 < \cdots < i_\ell \leqslant |E_n| \\ \text{for all } j \text{ and } k, \text{ if } j \in v_n \text{ and } k \notin v_n, \text{ then } f(E_{n,j}) < f(E_{n,k}) \end{cases}$$

  Update the internal state: $I_n \leftarrow \text{update}(I_{n-1}, E_n, v_n)$
  $S_n \leftarrow (E_{n,i_1}, \ldots, E_{n,i_\ell})$
  $x^{(f)}_{\omega,n} \leftarrow \text{proposal}(I_n)$
end while

We present SB-(μ, λ)-algorithms (resp. SB-$(\mu + \lambda)$-algorithms) in Algorithm 1.1. In this algorithm (also in Algorithm 1.2), the concatenation of the two vectors $x = (x_1, \ldots, x_k)$ and $x' = (x'_1, \ldots, x'_\ell)$ is denoted by $x \oplus x' = (x_1, \ldots, x_k, x'_1, \ldots, x'_\ell)$; we also use the shortcut $v \in (x_1, \ldots, x_k)$ to express that there exists $i \in [[1, k]]$ such that $x_i = v$.

Let us detail how the selection step is performed. If $E_n = (z_1, \ldots, z_p)$ is the vector of points considered at step n (either $E_n = O_n$ in the case of SB-(μ, λ)-ES or $E_n = O_n \oplus S_{n-1}$ in the case of SB-$(\mu+\lambda)$-ES), the function *select* returns a vector of μ distinct integers $v_n = (i_1, \ldots, i_\mu)$ such that:

$$\begin{cases} i_1 < \ldots < i_\mu \\ \text{for all } j \in \{i_1, \ldots, i_\mu\} \text{ and } k \in [[1, p]] \setminus \{i_1, \ldots, i_\mu\},\ f(z_j) < f(z_k) \end{cases}$$

Notice that the length of the vector v_n is equal to μ except maybe during the first iterations of the algorithm in the case $\lambda < \mu$: this explains the use of the auxiliary variable ℓ in Algorithm 1.1.

Algorithms with the "+" are usually termed *elitist*; this means that

we always keep the best individuals. Algorithms with the "," are termed *non-elitist*.

Finally, we would like to explain a generalization of SB-($\mu \overset{+}{,} \lambda$)-ES, called Full Ranking ($\mu \overset{+}{,} \lambda$)-ES (FR-($\mu \overset{+}{,} \lambda$)-ES). Instead of just giving the best μ points (i.e., the μ points with the lowest fitness values), we can consider a selection procedure which returns the best μ points *ordered with respect to their fitness*.

Algorithm 1.2 (Fournier and Teytaud, 2008) Full Ranking (μ, λ)-ES (resp. Full Ranking $(\mu + \lambda)$-ES). Framework for evolution strategies based on full ranking, working on a fitness function f. The random variable ω is a random seed. Compared to Algorithm 1.1, the vector of integers v_n obtained at the selection step is now ordered with respect to the fitness values of points from E_n; this framework is thus more general.

Initialization: $I_0 \leftarrow$ initial_state(ω); $S_0 \leftarrow \emptyset$; $n \leftarrow 0$
while true **do**
  $n \leftarrow n + 1$
  Generation step (generate an offspring O_n of λ distinct points)**:** $O_n \leftarrow$ generate(I_{n-1})
  $E_n \leftarrow O_n$ (resp. $E_n \leftarrow O_n \oplus S_{n-1}$)
  $\ell \leftarrow \min(\mu, |E_n|)$
  Selection step: $v_n \leftarrow \text{select}(E_n, f)$, i.e.:
    The vector $v_n = (i_1, \ldots, i_\ell)$ is defined by:

$$\begin{cases} i_1, \ldots, i_\ell \in [[1, |E_n|]] \\ f(E_{n,i_1}) < f(E_{n,i_2}) < \cdots < f(E_{n,i_\ell}) \\ \text{for all } j \notin v_n,\ f(E_{n,i_\ell}) < f(E_{n,j}) \end{cases}$$

  Update the internal state: $I_n \leftarrow \text{update}(I_{n-1}, E_n, v_n)$
  $S_n \leftarrow (E_{n,i_1}, \ldots, E_{n,i_\ell})$
  $x^{(f)}_{\omega,n} \leftarrow \text{proposal}(I_n)$
end while

The outline of these algorithms is summarized in Algorithm 1.2. More precisely, the selection step described in this algorithm works as follows. Given the vector of points $E_n = (z_1, \ldots, z_p)$ considered at step n, the function *select* returns a vector of μ distinct integers $v_n = (i_1, \ldots, i_\mu)$ such that:

$$\begin{cases} f(z_{i_1}) < \ldots < f(z_{i_\mu}) \quad \text{(the difference is here)} \\ \text{for all } j \in [[1,p]] \setminus \{i_1, \ldots, i_\mu\},\ f(z_{i_\mu}) < f(z_j). \end{cases}$$

(Once again, the length of the vector v_n may not be equal to μ at the beginning of the algorithm.)

Note that both Algorithms 1.1 and 1.2 define a class of algorithms: in order to obtain an algorithm, one has to specify the distribution of ω, how the offspring is generated (function *generate*), the space of states $\mathcal{I}$ as well as the functions *initial_state* and *update*, and finally the function *proposal*. Throughout the chapter, we assume that all functions involved in these algorithms are measurable.

Importantly, in spite of the formulation in Algorithms 1.1 and 1.2, the formalization is not restricted to deterministic algorithms. In Algorithms 1.1 and 1.2, all steps are deterministic, but there is an initial source of randomization, ω, which can be as large as required, e.g. it might be an infinite sequence of realizations of a random Gaussian variable or random uniform variables. So this is not a restriction, the algorithm must just be rephrased accordingly, so that ω is reported in the internal state and used in *e.g.* the "proposal" function.

1.3 INFORMAL SECTION: HOW TO DEFINE CONVERGENCE RATES?

There are algorithms which are termed quadratic, superlinear, or linear. Roughly speaking, quadratic means that the number of exact digits doubles at each iteration; superlinear means that the number of exact digits increases by a number which goes to infinity with iterations (but does not necessarily doubles); linear means that the number of exact digits increases by a (on average) constant number per iteration. The convergence order is 2 in the quadratic case, 1 in the linear case.

In the linear case, the convergence rate is the minimum constant c such that the distance to the optimum is upper bounded by $O(c^n)$, where n is the number of iterations. The convergence rate 0 is reserved for superlinear algorithms (e.g. BFGS).

These definitions are not so easy as it seems at first view. For example, what is an iteration? Should we consider each iterate? This would imply that we are not allowed to have an exploration step sometimes; each point far from the optimum "kills" the convergence rate. If the algorithm is very fast, and uses one random exploration step (uniform in the search space) once per 10 steps, then it is not even linear because of these random diversifications. Definitions can be modified in order to include such cases: we can distinguish between the points at which we compute the fitness, and the points which are proposed to the user as approximations of the

optimum; only the latter is considered when considering the distance to the optimum, and only the former for counting the number of iterations.

Other troubles remain: for which families of fitness functions does the convergence holds? In order to make these points clear, a simple solution is to define the supremum, over all fitness functions in a given family, of the number n of visited points such that the algorithm finds the optimum with precision ε.

However, many algorithms use random numbers. Therefore, we should consider the number n of points such that the algorithm finds the optimum with precision ε and with probability at least $1 - \delta$; this number will be termed hitting time and denoted by $n_{\epsilon,\delta}$.

Normalizations matter. In many cases, $n_{\epsilon,\delta}$ depends linearly in the dimension; therefore, we will consider normalizations so that a measure of convergence is the same for any dimensionality. This is the reason for our definition or convergence ratio, below. Incidentally, this definition will have the advantage that it can be used both for discrete and for continuous domains. We will see that recovering the convergence rate from the convergence ratio is not difficult.

1.4 BRANCHING FACTOR AND CONVERGENCE SPEED

We consider a bounded domain $D \subset \mathbb{R}^d$ and a norm $\|\cdot\|$ on $\mathbb{R}^d$. For $\varepsilon > 0$, we define $N(\varepsilon)$ to be the maximum integer n such that there exist n distinct points $x_1, \ldots, x_n \in D$ with $\|x_i - x_j\| \geqslant 2\varepsilon$ for all $i \neq j$. $N(\varepsilon)$ is termed the packing number, and quantifies the "size" and "dimensionality" of the domain as explained below. In particular, $N(\varepsilon) = |D|$ when ε is small enough in the case of a finite domain D, and $\log N(\varepsilon) \sim d \log(1/\varepsilon)$ when $\varepsilon \to 0$ if the domain D is bounded with non-empty interior (Kolmogorov and Tikhomirov, 1961).

Please notice that these definitions are consistent both for continuous domain $D \subset \mathbb{R}^d$ and discrete domain $D \subset \mathbb{R}^d$.

If each function $f \in \mathcal{F}$ has one and only one optimum $x^*(f) \in D$, for any given optimization algorithm as in Algorithm 1.2, and for $\varepsilon > 0$ and $\delta > 0$, we define $n_{\varepsilon,\delta}$ be the minimum number n of iterations such that with probability at least $1 - \delta$, an optimum is found at the n-th iteration within distance ε; i.e., $n_{\varepsilon,\delta}$ is minimal such that for all $n \geqslant n_{\varepsilon,\delta}$ and for all

$f \in \mathcal{F}$,

$$Pr_\omega[\|x^{(f)}_{\omega,n} - x^*(f)\| < \varepsilon] \geqslant 1 - \delta.$$

In order to state lower bounds on the convergence speed of evolution strategies obtained by (Fournier and Teytaud, 2008; Teytaud and Gelly, 2006), we first need to introduce a couple of definitions. Consider an algorithm of type $(\mu \dagger \lambda)$-ES working over a set of fitness functions $\mathcal{F}$. Let us define $L_n(\omega)$, the number of different paths followed by the algorithm on the random seed ω after n steps of computation when the function f runs over $\mathcal{F}$. More precisely,

$$L_n(\omega) = |\{(I_0, I_1, \ldots, I_n) \ : \ f \in \mathcal{F}\}|,$$

where the states I_i in the sequence above implicitly depend on both the function f and the random seed ω.

The *branching factor* K of this algorithm is defined as

$$K = \sup_E |\{\text{select}(E, f) \ : \ f \in \mathcal{F}\}|,$$

where the supremum holds for:

- E any set of λ in the case of SB-(μ, λ)-ES or Full Ranking (μ, λ)-ES;
- E any set of $\mu + \lambda$ in the case of SB-$(\mu + \lambda)$-ES or Full Ranking $(\mu + \lambda)$-ES.

The crucial idea around the concept of branching factor is that the number of different paths followed by some algorithm can be bounded in terms of the branching factor as follows:

$$L_n(\omega) \leqslant K \cdot L_{n-1}(\omega)$$

for all $n > 0$. Since the number of different points proposed after n steps of computation – denoted by $x^{(f)}_{w,n}$ in Algorithms 1.1 and 1.2 – is bounded by $L_n(\omega)$, the algorithm should converge slowly if $L_n(\omega)$ is small. This is quantified by the following result from [Teytaud and Gelly (2006)], restricted here to our purpose, relating the convergence speed to the branching factor of a $(\mu \dagger \lambda)$-algorithm.

Theorem 1.1 (Lower bound on the hitting time of $(\mu \dagger \lambda)$-ES.). *Consider a Full Ranking (μ, λ)-ES or $(\mu + \lambda)$-ES, as defined in Algorithm 1.2, and a set $\mathcal{F}$ of fitness functions on domain D, i.e. F is a set of functions from D to $\mathbb{R}$, i.e. $\mathcal{F} \subset \mathbb{R}^D$, such that any fitness function $f \in \mathcal{F}$ has only one min-argument $x^*(f)$, and such that $\{x^*(f) \ : \ f \in \mathcal{F}\} = D$. Let $\varepsilon > 0$ and $\delta \in]0, 1[$. Let $L_n(\omega)$ be the number of different paths (when*

the function f runs over $\mathcal{F}$) followed by the algorithm on the random seed ω. Then

$$E_\omega[L_{n_{\varepsilon,\delta}}(\omega)] \geqslant (1-\delta)N(\varepsilon).$$

In particular, if K denotes the branching factor of the algorithm, then

$$n_{\varepsilon,\delta} \geqslant \frac{\log(1-\delta)}{\log K} + \frac{\log N(\varepsilon)}{\log K}.$$

We point out that the assumptions on $\mathcal{F}$ are very mild; essentially we only assume that any point in the domain is a possible optimum.

We now define a quantity capturing the convergence speed of evolution strategies. Our aim is to have a unifying notion for both discrete and continuous domains, with a quantity analogous to the bounds of the form $O(1/d)$ established by (Witt and Jägersküpper, 2005) when $D \subset [0,1]^d$. We call *convergence ratio* of an algorithm for precision ε and parameters (μ, λ) the following:

$$\mathrm{CR}_\varepsilon^{(\mu,\lambda)} = \frac{\log N(\varepsilon)}{d n_{\varepsilon,\frac{1}{2}}}.$$

This quantity is very related to convergence rate - however, it is defined also in discrete cases, in which convergence rate does not make sense. People who are familiar with convergence rate will recover it by Formula 1.1 when both quantities exist.

$$-\log(\textit{convergence rate}) = \mathrm{CR}_\varepsilon^{(\mu,\lambda)} \tag{1.1}$$

It is known (Auger, 2005) that some evolution strategies lead to

$$\lim_{\varepsilon \to 0} \mathrm{CR}_\varepsilon^{(\mu,\lambda)} = O(1/d)$$

as $d \to \infty$, for a fixed μ and λ, e.g. on the sphere function or OneMax[4]. Note that the convergence ratio increases when the algorithm converges faster. Hence, a lower bound on the number of iterations of an algorithm for a given precision and a given confidence corresponds to an upper bound on its convergence ratio.

The ratio $\mathrm{CR}_\varepsilon^{(\mu,\lambda)}$ is relevant in the parallel setting, i.e. when working on a parallel computer, with parallel evaluation of the offspring. In the sequential setting (i.e. when individuals are evaluated sequentially), one should consider the *normalized convergence ratio* (normalized by the number of individuals generated per iteration) defined by $\mathrm{NCR}_\varepsilon^{(\mu,\lambda)} = \mathrm{CR}_\varepsilon^{(\mu,\lambda)}/\lambda$. In

[4]Some authors (but not all) consider evolution strategies only on continuous domains; however, our definition does not forbid discrete cases and our results include this case.

this chapter we will provide explicit statements on $\mathrm{NCR}_{\varepsilon}^{(\mu,\lambda)}$. Under some approximation, it has been shown that $\mathrm{NCR}_{\varepsilon}^{(\mu,\lambda)}$ is roughly proportional to $\log(\lambda)/\lambda$ for $\mu = 1$, and roughly independent of λ for μ freely chosen (Beyer, 1995). We will here show that the latter is true only in the case of λ small in front of the dimension and we will precisely quantify this and its relation with parallel speed-up, both in the continuous and the discrete cases. Theorem 1.1 can be rewritten in terms of the convergence ratio as follows. Consider a $(\mu \dot{+} \lambda)$-ES satisfying the hypothesis of Theorem 1.1. Let $\alpha(\varepsilon) = (1 - 1/\log N(\varepsilon))^{-1}$ (we shall use this notation throughout the chapter.) Then

$$\mathrm{CR}_{\varepsilon}^{(\mu,\lambda)} \leqslant \frac{\log K}{d} \cdot \alpha(\varepsilon), \tag{1.2}$$

and of course $\mathrm{NCR}_{\varepsilon}^{(\mu,\lambda)} \leqslant \frac{\log K}{d\lambda} \cdot \alpha(\varepsilon)$. This type of equation will be helpful for expressing results in a unified manner.

1.5 RESULTS IN THE GENERAL CASE

In this section, directly inspired by [Teytaud and Gelly (2006)], we do not assume anything on the fitness function. The results are therefore very general.

Formalizing the complexity of optimization algorithms is not a trivial task. In some cases, optimization is termed *expensive*: this means that the computational cost lies essentially in the evaluations of the fitness functions. Then, the cost is the number of fitness evaluations, or possibly the number of iterations in the case of parallel optimization (when all the population is evaluated simultaneously). The other extremal case is when the fitness function is very cheap - then, the internal cost of the algorithm has to be taken into account. A particular lower bound on the internal cost of the algorithm is the number of comparisons performed; the following theorem therefore lower bounds the number of comparisons necessary for reaching a given precision with a given confidence; the next subsection will consider the case of lower bounds on the number of fitness evaluations.

1.5.1 *LOWER BOUND ON THE NUMBER OF COMPARISONS*

As pointed out above, the number of comparisons is always a lower bound on the computational cost; if the cost of the fitness function is low and if the internal cost of the optimization algorithm is low, then this becomes the main computational cost. There are other cases in which the number of comparisons is a very natural criterion: when it is the only available information, as *e.g.* when parametrizing a strategy for two-players games. The following result is an immediate consequence of Theorem 1.1.

Theorem 1.2 (Convergence rate w.r.t the number of comparisons). *Assume the same hypothesis as in Theorem 1.1 with $K = 2$ corresponding to the result of a comparison between the fitnesses of two previously visited points: SB-(μ, λ)-ES* with $\mu = 1$, $\lambda = 2$. *Importantly, this is not restricted to algorithms usually formalized with $\mu = 1$, $\lambda = 2$:* all *algorithms based on comparisons can be rewritten under this form, with one iteration of the algorithm (with population size 2) for each comparison (the population is the couple of points to be compared).*

Then, with $\log_2(x) = \log(x)/\log(2)$*, the number of comparisons n_c required for ensuring with probability $1 - \delta$ a precision ε is $n_c \geqslant \log_2(1 - \delta) + \log_2(N(\varepsilon))$. I.e., formally, $P(\|x_{n_c} - x^*(f)\| < \varepsilon) \geqslant 1 - \delta \Rightarrow n_c \geqslant \log_2(1 - \delta) + \log_2(N(\varepsilon))$.*

This bound, leading to $\lim_{\varepsilon \to 0} \mathrm{CR}_\varepsilon{}^{(1,2)} = O(1/d)$ is tight: e.g. for $D = [0, 1]^d$, $\mathcal{F} = \{x \mapsto \|x - x^*(f)\|_1; x^*(f) \in [0, 1]^d\}$ is solved with convergence ratio $\Omega(1/d)$ by Algorithm 1.3 (close to the Hooke&Jeeves algorithm by [Hooke and Jeeves (1961)]).

1.5.2 *LOWER BOUND ON THE NUMBER OF FUNCTION EVALUATIONS*

We already mentionned that our approach covers almost all existing evolutionary algorithms. We can now check the value of K, depending on the algorithm, and consider convergence rates with respect to the number of fitness-evaluations instead of the number of comparisons. The convergence ratio will verify Eq. 1.2.

- (μ, λ)**-ES (or SA-ES) :** at each step, then, we only know which are the selected points. Then, $K = \binom{\lambda}{\mu} \leqslant \binom{\lambda}{\lfloor \lambda/2 \rfloor} \leqslant (2^\lambda / \sqrt{2\pi\lambda})$

Algorithm 1.3 A simplified version of the Hooke&Jeeves algorithm, matching the bound in Theorem 1.2. It is a $(1+1)$-ES (the difference between SB and FR does not make sense in $(1+1)$-ES).

Initialize $x = (x_1, \ldots, x_d)$ at any point.
for In lexicographic order on $(j, i) \in \mathbb{N} \times [[0, d-1[[$ **do**
  Try to replace the j^{th} bit b of x_i by $1-b$; let x' be the result.
  Evaluate the fitness at x'
  if x' is better than x **then**
    $x = x'$
  end if
end for

(**?**)[p587]DEV or (**?**)[]feller68 for classical properties of $\binom{\lambda}{\mu}$).

- Consider **more generally, any selection based algorithm**, i.e. any algorithm in which v_n encodes only a subset of μ points among λ. Then, the algorithm provides only a subset of $[[1, \lambda]]$, i.e. $K \leqslant 2^\lambda$ (even if μ is not fixed and chosen by the algorithm at each iteration). Therefore, $\mathrm{CR}_\varepsilon^{(\mu,\lambda)} \leqslant \frac{\lambda}{d} \cdot \alpha(\varepsilon)$. This bound can be outperformed by full-ranking algorithms as shown by [Teytaud and Gelly (2006)]; this shows that in some cases, full-ranking can significantly (more than of a constant factor only) outperform selection. However, we will see that this does not hold anymore under some mild assumptions, at least for (μ, λ)-ES.
- $(\mu+\lambda)$**-ES:** then, we only know which are the selected points. Then, $K = (\lambda+\mu)!/(\mu!\lambda!)$. This does not allow a proof of $\mathrm{CR}_\varepsilon^{(\mu,\lambda)} = \Theta(\lambda/d)$ if μ increases as a function of d. This point will be further analyzed using VC-dimension in the next section; essentially the ultimate rates for elitist strategies $(+)$ are an open problem.

[Teytaud and Gelly (2006)] also show that superlinear convergence rates are possible, in some very specific cases, when using the full ranking information. Next section will show that this is not possible for more reasonnable fitness functions (as shown in Table 1.1).

1.6 BOUNDS ON THE CONVERGENCE SPEED USING VC-DIMENSION

Lower bounds on the convergence speed of evolutions strategies can be obtained from Equation 1.2 (Section 1.4) as soon as an upper bound on the branching factor K is known. In the results above, we used essentially elementary combinatorial properties: in particular, the fact that the number of subsets of size μ of a set of λ points (and thus the branching factor K) is at most $\binom{\lambda}{\mu} \leqslant \binom{\lambda}{\lfloor \lambda/2 \rfloor}$. This surely holds, but it is a worst case on all possible selections. If we have the additional hypothesis that the fitness functions are *reasonnable*, many subsets of size μ of a set of size λ cannot be realized by a selection step. This is precisely quantified by the theory of VC-dimension and the shatter function lemma (also known as Sauer's lemma). In this section, we show how a VC-dimension hypothesis allows to obtain improved lower bounds on the convergence speed of $(\mu \dot{+} \lambda)$-ES. VC-dimension is the central tool of many works, in particular for learning theory (Vapnik and Chervonenkis, 1968).

Given a function f defined over D and $r \in \mathbb{R}$, let $B_{f,r} = \{x \in D \ : \ f(x) < r\}$. We define the *sublevel sets* $L_{\mathcal{F}}$ of a set $\mathcal{F}$ of functions defined over the domain D as

$$L_{\mathcal{F}} = \{B_{f,r} \ : \ f \in \mathcal{F}, \ r \in \mathbb{R}\}.$$

This section presents lower bounds on the convergence speed of algorithms optimizing $\mathcal{F}$ in terms of the VC-dimension of the sublevel sets $L_{\mathcal{F}}$. The results were originally proved by [Fournier and Teytaud (2008)].

We now briefly recall the definition of VC-dimension and the shatter function lemma (Vapnik and Chervonenkis, 1971; Sauer, 1972) – our presentation is based on the work by [Matoušek (2002)]. A set system on a set A is a family $\mathcal{S}$ of subsets of A. For $B \subseteq A$, we define the restriction of $\mathcal{S}$ to B as $\mathcal{S}|_B = \{S \cap B \ : \ S \in \mathcal{S}\}$. The VC-dimension of the set system $\mathcal{S}$ defined over A is defined as $\sup\{|B| \ : \ \mathcal{S}|_B = 2^B\}$ where 2^B denotes the power set of B; in other words, it is the size of the largest subset B of A such that any subset of B can be obtained by intersecting B with an element of $\mathcal{S}$. Given a set system $\mathcal{S}$ over A, the *shatter function* $\pi_{\mathcal{S}}$ is defined by $\pi_{\mathcal{S}}(m) = \max\{|\mathcal{S}|_B| \ : \ B \subseteq A, \ |B| = m\}$; thus $\pi_{\mathcal{S}}(m)$ is the maximum number of different subsets of A which can be obtained by intersecting a single subset of size m of A with all elements of $\mathcal{S}$. We next recall the following lemma which gives an upper bound on $\pi_{\mathcal{S}}$ in terms of the VC-dimension of $\mathcal{S}$.

Lemma 1.1 (Shatter function lemma). *For any set system $\mathcal{S}$ of VC-dimension d, then for all integer m, it holds that $\pi_{\mathcal{S}}(m) \leqslant \sum_{i=0}^{d} \binom{m}{i}$.*

At last, let us recall the following classical bound (Devroye *et al.*, 1997) which is valid whenever $d \geqslant 3$:

$$\sum_{i=0}^{d} \binom{m}{i} \leqslant m^d. \tag{1.3}$$

Note that the trivial bound $\sum_{i=0}^{d} \binom{m}{i} \leqslant 2^m$ is tight when $m \leqslant d$. The interesting case happens when m is large with respect to the VC-dimension d: the bound stated in Equation 1.3 becomes polynomial in m in this case.

We now give a couple of examples of bounded VC-dimension for some classical set systems in $\mathbb{R}^d$ (Devroye *et al.*, 1997). Axis-parallel hyper-rectangles have a VC-dimension bounded by $2d$. Sphere functions in $\mathbb{R}^d$ (for the Euclidean norm) have a VC-dimension equal to $d+1$. The VC-dimension of ellipsoids in $\mathbb{R}^d$ is bounded by $d(d+1)/2+d$. More generally, subsets of $\mathbb{R}^d$ defined by polynomial inequalities involving a finite number of real parameters, and Boolean combinations of these, have a finite VC-dimension (Matoušek, 2002) (Chapter 10.3). Also, any set of functions which is the set of translations of a given function has finite VC-dimension.

In the rest of this section, we shall state bounds for set systems of VC-dimension at least 3. However, the case of VC-dimension smaller than 3 can be handled in a similar way: the bound above has to be replaced with $\sum_{i=0}^{d} \binom{m}{i} \leqslant m^d + 1$.

Upper bounds on the convergence ratio of evolution strategies obtained in this section are summarized and compared to results of the previous section in Table 1.1 (we recall that $\alpha(\varepsilon) = (1 - 1/\log N(\varepsilon))^{-1}$).

Hence, any algorithm of type $(\mu \stackrel{+}{,} \lambda)$-ES optimizing a set of fitness functions with sublevel sets of VC-dimension $\leqslant V$ on a domain $D \subset \mathbb{R}^d$ has a convergence ratio which satisfies the bounds given in Table 1.1. Let us remark that the sphere functions, i.e., the set of fitness functions $\mathcal{F} = \{f_c : c \in \mathbb{R}^d\}$ where $f_c(x) = \|x - c\|_2$, are a special case of functions with spheres as sublevel sets, but are not the only ones. (The same remark applies to hyper-rectangles and ellipsoids.)

We now derive specific results for various families of ES: non-elitist strategies (μ, λ)-ES, non-elitist full ranking ES, and elitist $(\mu + \lambda)$-ES.

1.6.1 *SELECTION-BASED NON-ELITIST EVOLUTION STRATEGIES*

Theorem 1.3 (Selection Based (μ, λ)-ES). *Consider an SB-(μ, λ)-ES (Algorithm 1.1) in a domain $D \subset \mathbb{R}^d$, such that $D = \{x^*(f) \ : \ f \in \mathcal{F}\}$. Let $V \geqslant 3$ be the VC-dimension of the sublevel sets of $\mathcal{F}$. The convergence ratio of this algorithm satisfies*

$$\mathrm{CR}_\varepsilon^{(\mu,\lambda)} \leqslant \frac{V \log \lambda}{d} \cdot \alpha(\varepsilon),$$

or equivalently

$$n_{\varepsilon,\frac{1}{2}} \geqslant \frac{\log N(\varepsilon)\alpha(\varepsilon)}{V \log(\lambda)}.$$

The detailed proof is given by [Fournier and Teytaud (2008)] and combines a VC-type bound and Equation 1.2 from Theorem 1.1.

Remark. If $V = 1$ or $V = 2$, then the bound obtained in Theorem 1.3 becomes $\mathrm{CR}_\varepsilon^{(\mu,\lambda)} \leqslant \frac{V \log(\lambda+1)}{d} \cdot \alpha(\varepsilon)$. The case $V < 3$ can be handled in a similar way throughout the chapter: $V \log \lambda$ is replaced with $V \log(\lambda + 1)$ in this case.

The bound obtained in Theorem 1.3 is interesting when λ is large, and μ not too close to 1 or λ. Indeed, the trivial bound $K \leqslant \binom{\lambda}{\mu}$ combined with Equation 1.2 as above, yields better bounds on the convergence ratio in these cases.

We now briefly consider Full Ranking (μ, λ) evolution strategies. That is, we move from Algorithm 1.1 to the more general setting of Algorithm 1.2.

1.6.2 *FULL RANKING (μ, λ)-ES WITH μ NOT TOO LARGE*

For evolutions strategies of type Full Ranking (μ, λ)-ES, an upper bound on the number of possible outcomes of the selection step (including the ranking of children) is obtained by multiplying by $\mu!$ the number of possible outcomes in the case of selection only. This multiplies the branching factor K accordingly and leads to

$$\mathrm{CR}_\varepsilon^{(\mu,\lambda)} \leqslant \frac{1}{d}(V \log(\lambda) + \mu \log \mu) \cdot \alpha(\varepsilon).$$

However, we can say better in the case where μ is large with respect to the VC-dimension V of the sublevel sets of the fitness functions; this will be the subject of the next section.

Application to optimization of the OneMax function

We end this section with an application by [Fournier and Teytaud (2008)] to optimization in the discrete domain $D = \{0,1\}^d$. For ε sufficiently small, the balls are singletons; it follows that $N(\varepsilon) = N(0) = 2^d$ and $\alpha(\varepsilon) = \alpha(0) = 1/(1-1/d)$ when ε is small enough and $d \geqslant 2$. Let us consider the OneMax function defined by $x \longmapsto \sum_{i=1}^{d} x_i$, and all its symmetries; the set of fitness functions is

$$\mathcal{F} = \{f_\eta : x \longmapsto \sum_{i \in [[1,d]]} |x_i - \eta_i|,\ \eta \in \{0,1\}^d\}.$$

Classical results around OneMax are as follows:

- a $(1+1)$-ES finds the optimum after $\Theta(d\log(d))$ iterations on average[5];
- all black-box algorithms need average computation time at least $\Theta(d\log(d))$.

Let us discuss the case of Selection based (μ,λ)-ES for this family of functions. The main strength of results below is that we improve these lower bounds as a function of λ for the whole family of selection-based evolutionary algorithms. The aim is to find the point where this function is maximum (hence, the selection step of an $(\mu \overset{+}{,} \lambda)$-ES keeps μ points with *highest* fitness values). Notice that no $(\mu \overset{+}{,} \lambda)$-ES can avoid to generate points with equal fitness values in a single step as soon as $\lambda \geqslant 3$. Indeed, given three different points of $\{0,1\}^d$, there must be two of them x and y such that the Hamming weight of the symmetric difference of x and y is even; then there exists η such that $f_\eta(x) = f_\eta(y)$.

The VC-dimension of sublevel sets of $\mathcal{F}$ satisfies $V \leqslant d+1$ ()[chapter 13]DEV. Therefore, the convergence ratio $\mathrm{CR}_\varepsilon^{(\mu,\lambda)}$ is at best, for ε sufficiently small, $2(1-1/d^2)\log\lambda$ in the case of Selection based (μ,λ)-ES. Bounds on the convergence ratio of other kinds of strategies are obtained in the same way. As well as in many cases for the continuous domain, we get $O(\log(\lambda))$ independently of the dimension; this is a consequence of the definition of $\mathrm{CR}_\varepsilon^{(\mu,\lambda)}$.

The bound on the convergence ratio obtained above yields the following runtime for solving OneMax in dimension d with probability $1/2$:

$$\begin{cases} n_{0,\frac{1}{2}} = \Omega\left(1/\log(\lambda)\right) & \text{if } \lambda/\log\lambda \geqslant d, \text{ (degenerate bound)} \\ n_{0,\frac{1}{2}} = \Omega\left(d/\lambda\right) & \text{if } \lambda/\log\lambda < d. \end{cases} \tag{1.4}$$

[5]This is realized by a mutation in which each bit is flipped independently with probability $1/d$.

The case where $\lambda \leqslant 2d$ above is obtained by Equation 1.2, using the trivial bound $K \leqslant \binom{\lambda}{\lambda/2} \leqslant \binom{2d}{d}$.

We can give some elements of tightness in the (quite strong) model where the algorithm has access to the whole set of points in $\{x \in E_n; \forall y \in E_n, OneMax(x) \leqslant OneMax(y)\}$; this algorithm has access to the set of points with optimal fitness in E_n, of size possibly larger than μ in case of equality. Indeed, Equation 1.4 is tight in this model in the special cases $\lambda = d + 1$ and $\lambda = O(1)$ (independent of d); the proof is omitted for short.

For the sake of parallelization of OneMax solving with ES, this suggests the use of λ linear in the dimension d; at least for optimally parallel comparison-based algorithms. Indeed, for a parallel evaluation of the λ generated points, we get an almost linear speed-up for $\lambda \leqslant d$ – the convergence ratio is $\Omega(1)$ for $\lambda = d + 1$ whereas it is $O(1/d)$ for $\lambda = O(1)$ – while the speed-up is asymptotically logarithmic in λ when $\lambda \to \infty$.

1.6.3 *FULL RANKING* (μ, λ)*-ES WITH* μ *LARGE*

Using a lemma by (Fournier and Teytaud, 2008) around the number of possible rankings of λ points by functions with finite VC-dimension, the following theorem provides an upper bound on the number of possible orders of a fixed set of points with respect to fitness functions $f \in \mathcal{F}$, when the sublevel sets of $\mathcal{F}$ have a finite VC-dimension.

Theorem 1.4 (Full Ranking (μ, λ)-ES). *Consider a (μ, λ)-ES (Algorithm 1.2) in a domain $D \subset \mathbb{R}^d$, such that $D = \{x^*(f) \ : \ f \in \mathcal{F}\}$. Let $V \geqslant 3$ be the VC-dimension of the sublevel sets of $\mathcal{F}$. The convergence ratio of this algorithm satisfies*

$$\mathrm{CR}_{\varepsilon}^{(\mu,\lambda)} \leqslant \frac{V\,(4\mu + \log\lambda)}{d} \cdot \alpha(\varepsilon),$$

or equivalently

$$n_{\varepsilon,\frac{1}{2}} \geqslant \frac{\log(N(\varepsilon)\alpha(\varepsilon)}{V\,(4\mu + \log\lambda)}$$

An important point here is the μ out of the logarithm. We will see that in some cases (typically the sphere function) we can do better.

1.6.4 *ELITIST CASE:* $(\mu + \lambda)$*-ES*

The elitist case (full ranking or not) is a corollary of Theorem 1.3.

Corollary 1.1 (Selection Based $(\mu+\lambda)$-ES and Full Ranking $(\mu+\lambda)$-ES). *Let $\mathcal{F}$ be a set of fitness functions defined in a domain $D \subset \mathbb{R}^d$, such that $D = \{x^*(f) \ : \ f \in \mathcal{F}\}$. Let V be the VC-dimension of the sublevel sets of $\mathcal{F}$. The convergence ratio of an algorithm of type Selection Based $(\mu+\lambda)$-ES for $\mathcal{F}$ satisfies:*

$$\mathrm{CR}_{\varepsilon}^{(\mu,\lambda)} \leqslant \frac{V \log(\mu+\lambda)}{d} \cdot \alpha(\varepsilon),$$

or equivalently

$$n_{\varepsilon,\frac{1}{2}} \geqslant \frac{\log(N(\varepsilon))\alpha(\varepsilon)}{V \log(\mu+\lambda)}.$$

For an algorithm of type Full Ranking $(\mu+\lambda)$-ES, the following holds:

$$\mathrm{CR}_{\varepsilon}^{(\mu,\lambda)} \leqslant \frac{V\,(4\mu + \log(\mu+\lambda))}{d} \cdot \alpha(\varepsilon),$$

or equivalently

$$n_{\varepsilon,\frac{1}{2}} \geqslant \frac{\log(N(\varepsilon))\alpha(\varepsilon)}{V\,(4\mu + \log(\lambda))}.$$

1.7 SIGN PATTERNS AND THE CASE OF THE SPHERE FUNCTION

This sections presents a technique proposed by [Fournier and Teytaud (2008)] based on the number of sign patterns to derive lower bounds on the convergence speed of full ranking algorithms. Applied to the special case of the sphere function, it shows that the speed-up is asymptotically at most logarithmic in λ. (See Proposition 1.1.) This improves on the bounds obtained by VC-dimension arguments in Theorem 1.4 which did not forbid a speed-up linear in μ.

Although it is applied to the sphere function, the technique used here applies to any system where the number of sign patterns can be efficiently bounded, such as quadratic functions with positive Hessian. In fact, polynomials of bounded degree are amenable to this technique – we refer the reader to the works by [Rónyai *et al.* (2001)] and [Matoušek (2002)] (chapter 6.2) for further details. However, as opposed to the previous section, the bound obtained by the sign pattern technique presented here does not apply to the general case of functions with spheres (or ellipsoids) as sublevel sets[6]. On the other hand, as all results in this chapter, they can be applied to any monotonic transformations of sphere functions.

[6]We recall that there are functions which have spheres as level sets (all level sets are spheres) without these sphere to be centered on the optimum.

For the sphere function, we point out in Section 1.7.2 that λ linear in the dimension provides an almost linear speed-up. Indeed, the straightforward parallelization performed by distributing the λ fitness evaluations over λ processors has an almost linear speed-up until at least λ equal to twice the dimension (while the speed-up is asymptotically at most logarithmic in λ by Proposition 1.1.)

1.7.1 *LOWER BOUNDS ON FULL RANKING STRATEGIES VIA THE NUMBER OF SIGN PATTERNS*

We present an alternative method to obtain improved lower bounds on the convergence speed of evolution strategies which use full ranking. For a real x, we define its sign to be $\text{sign}(x) = 0$ if $x = 0$, $\text{sign}(x) = 1$ if $x > 0$, and $\text{sign}(x) = -1$ if $x < 0$. Given a fitness function f and a finite set of points $x_1, \ldots, x_\lambda \in D$, we define the set of realizable sign conditions as

$$\text{Sign}_{f,(x_1,\ldots,x_\lambda)} = \{\text{sign}(f(x_i) - f(x_j)) : 1 \leqslant i < j \leqslant \lambda\}.$$

This method can be applied as soon as the number of sign conditions, i.e. the number of possible sign vectors, can be efficiently bounded. Indeed, the following inequality on the branching factor holds:

$$K \leqslant \max\left\{|\text{Sign}_{f,(x_1,\ldots,x_\lambda)}| : f \in \mathcal{F},\ x_1, \ldots, x_\lambda \in D\right\}. \tag{1.5}$$

We now apply the above remark to the sphere functions, in order to obtain an improved lower bound on the convergence speed of full ranking strategies for these functions. We recall that the sphere function is[7] in fact the the set of fitness functions $\mathcal{F} = \{f_c \ : \ c \in \mathbb{R}^d\}$ with $f_c(x) = \|x - c\|_2$ (where $\|\cdot\|_2$ denotes the Euclidean norm, i.e. $f_c(x) = ((x_1 - c_1)^2 + \ldots + (x_d - c_d)^2)^{1/2}$).

Proposition 1.1. *Let $d \geqslant 3$. Consider a Full Ranking (μ, λ)-ES, as in Algorithm 1.2, optimizing the sphere function in a domain $D \subset \mathbb{R}^d$. Then*

$$\text{CR}_\varepsilon^{(\mu,\lambda)} \leqslant 2\log(\lambda) \cdot \alpha(\varepsilon).$$

Remark. The case of equality (i.e., when it is possible for two generated points x and y to satisfy $f(x) = f(y)$) can be handled thanks to bounds on the number of faces of the hyperplanes arrangements, using the work by [Matoušek (2002)] (chapter 6.1).

[7]However, for comparison-based algorithms, replacing a function by its composition with an increasing function from $\mathbb{R}$ to $\mathbb{R}$ does not change the behavior of the algorithm; therefore, the sphere function is often extended to all its compositions with increasing functions.

1.7.2 *FAST CONVERGENCE RATIO WITH $\lambda = 2d$ FOR OPTIMIZING THE SPHERE FUNCTION*

We point out here that for the specific case of the sphere function, a convergence ratio of $\Theta(1)$ can be reached with $\mu = \lambda = 2d$ in the domain $[0,1]^d$ by some algorithm of type Full Ranking (μ, λ)-ES.

This convergence ratio is easily obtained with the following algorithm in the spirit of the Hooke and Jeeves algorithm (Hooke and Jeeves, 1961). Let e_i denote the vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ with a unique 1 in position i. First split $[0,1]^d$ into the 2^d cells delimited by the d hyperplanes of equations $x_i = 1/2$; the full ranking of the $2d$ points $\{(\frac{1}{2}, \frac{1}{2}, \ldots, \frac{1}{2}) + \frac{\eta}{2} e_i \ : \ 1 \leqslant i \leqslant n, \ \eta \in \{-1, 1\}\}$ allows to decide in which of these cells the optimum lies; then the algorithm proceeds recursively. (See Algorithm 1.4.)

Algorithm 1.4 Fast algorithm of type Full Ranking (μ, λ)-ES for the sphere function in the domain $[0,1]^d$ in the special case $\mu = \lambda = 2d$.

$x \leftarrow (1/2, \ldots, 1/2)$; $\sigma \leftarrow 1/2$
while true **do**
  Generate $\lambda = 2d$ distinct points equal to $x \pm \sigma e_i$
    (e_i denotes the vector $(0, \ldots, 0, 1, 0, \ldots, 0)$ with a unique 1 in position i)
  With the ranking information, decide in which octant Δ of $x + [-\sigma, \sigma]^d$ is the optimum
  Move x to the center of the octant Δ
  Set $\sigma \leftarrow \sigma/2$
end while

After n iterations, the point x_n proposed by this algorithm satisfies $\|x_n - x^*(f)\|_2 \leqslant \sqrt{d}/2^n$. Moreover, this distance is realized by some fitness function. It follows that $n_{\varepsilon, \frac{1}{2}} = \log \frac{1}{\varepsilon} + \frac{1}{2} \log d$. On the other hand $\log(N(\varepsilon)) = \Theta(d \log \frac{1}{\varepsilon})$. Thus, we have obtained an algorithm for the sphere function in dimension d which satisfies:

$$\mathrm{CR}_{\varepsilon}^{(2d,2d)} = \Theta(1). \tag{1.6}$$

Notice that for $\lambda = O(1)$, independent from d, the branching factor of any algorithm satisfies $K = O(1)$; it follows by Equation 1.2 that any algorithm optimizing the sphere function in dimension d satisfies $\mathrm{CR}_{\varepsilon}^{(\lambda,\lambda)} = O(1/d)$ in this case. Hence, Algorithm 1.4 achieves an almost linear speed-up $\Omega(d)$ when λ moves from $O(1)$ to $2d$. On the other hand, the asymptotic speed-up for λ large (and d fixed) is known to be at most logarithmic ($\mathrm{CR}_{\varepsilon}^{(\lambda,\lambda)} = O(\log(\lambda))$ by Proposition 1.1).

1.8 CONCLUSIONS AND FURTHER WORK

The main strength of the approach presented above is its generality: it can be applied to evolution strategies, direct search methods, cross-entropy methods, estimation of distribution algorithms. It shows that

Table 1.1 Upper bound on the convergence ratio. The first column is the general case (no assumption on the set of fitness functions except that the optimum can be anywhere in the domain). The second column is when the sublevel sets of fitness functions have VC-dimension V in $\mathbb{R}^d$. The third column is just the application of the second column to the case of convex quadratic functions ($V = \Theta(d^2)$). The fourth column is the special case of the sphere function, or, equivalently, any set of functions in which the sublevel sets are homotheties of a fixed ellipsoid.

Framework	General case	Bounded VC-dimension	Quadratic case	Sphere function
SB-(μ, λ)-ES	$\frac{1}{d}(\lambda \log(2) - \frac{1}{2}\log(2\pi\lambda))$	$\frac{V}{d}\log(\lambda)$	$O(d)\log(\lambda)$	$(1+\frac{1}{d})$ $\times \log(\lambda)$
SB-$(\mu + \lambda)$-ES	$\frac{1}{d}\log(\frac{(\lambda+\mu)!}{\lambda!\mu!})$	$\frac{V}{d}\log(\mu$ $+\lambda)$	$O(d)\log(\mu$ $+\lambda)$	$(1+\frac{1}{d})$ $\times \log(\mu+\lambda)$
FR-(μ, λ)-ES		$\frac{V}{d}(4\mu$ $+\log\lambda)$	$O(d)(4\mu+$ $\log\lambda)$	$2\log(\lambda)$
FR-$(\mu + \lambda)$-ES		$\frac{V}{d}(4\mu+$ $\log(\lambda+\mu))$	$O(d)(4\mu+$ $\log(\lambda+\mu))$	$(1+\frac{1}{d}) \times (4\mu$ $+\log(\lambda+\mu))$

- with respect to the number of comparisons, the convergence is at best linear with convergence rate $\exp(-O(1/d))$ where d is the dimension;
- with respect to the number of fitness evaluations and for sphere functions, the convergence is at best linear with convergence ratio $O((1+1/d)\log(\lambda))$ where d is the dimension; dividing by λ, this leads to $(1+1/d)\log(\lambda)/\lambda$ for λ large, and $\Omega(1)$ for $\lambda = d$. This suggests that choosing λ of the same order of the dimension does not significantly reduces the efficiency on a sequential computer. The classical trick consisting in increasing λ for avoiding local minima has therefore no significant cost from the point of view of the rate, even on a sequential computer.
- with respect to the number of iterations (typically the running time on a parallel computer when the internal cost of the algorithm is negligible and with one evaluation of individual per processing unit), the convergence, in most cases, is at best linear; the convergence ratio $O(p/d)$ for a number $p = \Theta(d)$ of computation units,

and then linear with convergence rate $O(\log(p)/d)$ for $p \gg d$; this is proved for bounded VC-dimension and SB non-elitist algorithms, and also for FR non-elitist strategies applied to the sphere fitness function.

The most important points, for the author, are: (i) the presence of μ out of the logarithm in some cases; (ii) the $O(d)$ for the quadratic case: it's likely that if the considered space of functions is translation invariant then this should be $O(1)$. This is already the case in the quadratic case, if the sublevel sets are homotheties of one and only one ellipsoid (becasue in this case the VC-dimension is the same as for the sphere).

We have illustrated the results mainly in the continuous domain (bounded open subsets of $\mathbb{R}^d$), a little on $\{0,1\}^d$; similar results have also been applied in multiobjective cases (Teytaud, 2007). However, the scope of the result is probably larger; the main strength of evolutionary algorithms is that they are robust and stable in structured spaces; it is likely that one can apply these bounds for *e.g.* sets of permutations, sets of trees, neural networks with their structures, etc.

A main implication of our results is the limit on the parallelization. In the case of evolution strategies based on selection only (algorithms of type SB-(μ,λ)-ES), the linear speed-up of selection based evolution strategies shown by (Beyer, 1995) cannot be obtained for λ large enough. Asymptotically, the speed-up obtained with such an algorithm is at most logarithmic as shown in Theorem 1.3. On the other hand, we show that the speed-up can be nearly linear for up to $2d$ processors on the sphere function in dimension d. A similar phenomenon is observed in the case of the discrete function OneMax.

The case of equality between fitness values of distinct points is not handled here; it is possible to show that, in many cases, the bound is just augmented by a constant factor (**?**)[]fournierAlgorithmica.

A somewhat puzzling point is that bounds on the convergence speed obtained by VC-dimension arguments are weaker when the function is more "complex" (i.e., when the VC-dimension of its sublevel sets is higher). This is not surprising after all: it is easier to have very fast algorithms on very strange fitness functions than on "natural" fitness functions; indeed, one can wonder if it is possible to build *ad hoc* fitness functions matching the bounds obtained by VC-dimension arguments. Such constructions were given by (Teytaud and Gelly, 2006) to match lower bounds on the conver-

gence speed of algorithms obtained from the sole branching factor[8] - this provides very strange fitness functions, of low practical interest but showing that improving the results requires some more assumption. For example, the lower bounds for quadratic fitness functions are improved (they become equivalent to sphere fitness functions) if we consider only quadratic fitness functions which are translations of only one quadratic fitness function.

Also we know that the bounds obtained from VC-dimension arguments can be far from optimal for a specific family of fitness functions (this does not imply that the bound is not tight in the general case of a given VC-dimension): the bound obtained for Full Ranking (μ, λ)-ES is greatly improved in the special case of the sphere function in Section 1.7.

A related question is the parallelization of optimization of ill-conditioned functions. Can the speed-up be linear until roughly $\lambda = d^2$ computation units?

When moving from algorithms of type Selection Based (μ, λ)-ES to Full Ranking (μ, λ)-ES, and in the general case of any set of fitness functions, the speed-up that can be achieved moves from logarithmic to linear in λ (we have only presented here only lower bounds for the general case, but [Teytaud and Gelly (2006)] have shown that they can be reached.). However, we know from Proposition 1.1 that the speed-up is at most logarithmic for a Full Ranking (μ, λ)-ES in the special case of the sphere function and this linear speed-up is therefore essentially theoretical – see also the discussion following Proposition 1.1. An important question is then the existence of important families of functions for which a linear speed-up can be achieved. The answer is probably yes within a setting adapted to highly multimodal cases (with much bigger hitting times however), and no in the "easy" case as in this paper (we need a family of functions which is more parallelizable than the sphere functions!).

A related intriguing question is the convergence ratio that can be reached for selection based algorithms (i.e., without keeping the full ranking information) for the sphere function. In particular, is it possible to achieve a constant convergence ratio with λ linear in the dimension, as in Equation 1.6? To the best of our knowledge, this is an open problem - preliminary investigation suggests that the answer is positive.

[8]More precisely, (Teytaud and Gelly, 2006) show that there is a FR algorithm a and a set of fitness function F so that a is "very" fast on F, and in particular much more than rates achieved by any algorithm on the sphere; this is of theoretical interest only as the corresponding algorithm is useless except for these specific families of functions.

Finally, the results can be extended beyond the comparison-based case. Fitness-based or gradient-based or Hessian-based algorithms are concerned also when taking into account the finite number bits of the answers from the oracle. In particular, it is known that theoretically quadratically converging algorithms are in fact linear due to finite precision. The finite number of bits is for sure a limitation which can be handled by bounds as in this chapter.

ACKNOWLEDGMENTS

We would like to thank Anne Auger and Fabien Teytaud for constructive talks. We point out also that Hervé Fournier contributed a lot to this work, and could almost be called a coauthor. This work was partially supported by the Pascal Network of Excellence. This work has been supported by French National Research Agency (ANR) through COSINUS program (project EXPLO-RA nANR-08-COSI-004). It was also supported by the FP7 program under the Mash project (grant 247022).

Bibliography

Arnold, D. V. (2005). Optimal weighted recombination, in *Foundations of Genetic Algorithms 8*, *Lecture Notes in Computer Science*, Vol. 3469 (Springer-Verlag, Berlin Heidelberg), ISBN 3-540-27237-2, pp. 215–237.

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

Auger, A., Schoenauer, M. and Teytaud, O. (2005). Local and global order 3/2 convergence of a surrogate evolutionary algorithm, in *GECCO '05: Proceedings of the 2005 conference on Genetic and evolutionary computation* (ACM, New York, NY, USA), ISBN 1-59593-010-8, pp. 857–864, doi:http://doi.acm.org/10.1145/1068009.1068154.

Bäck, T., Hoffmeister, F. and Schwefel, H.-P. (1991). Extended selection mechanisms in genetic algorithms, in R. K. Belew and L. B. Booker (eds.), *Proceedings of the Fourth International Conference on Genetic Algorithms* (Morgan Kaufmann Publishers, San Mateo, CA), pp. 92–99.

Baker, J. E. (1987). Reducing bias and inefficiency in the selection algorithm, in *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application* (Lawrence Erlbaum Associates, Inc., Mahwah, NJ, USA), ISBN 0-8058-0158-8, pp. 14–21.

Baluja, S. (1994). Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning,, Tech. Rep. CMU-CS-94-163, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

Beyer, H.-G. (1995). Toward a theory of evolution strategies: On the benefit of sex – the $(\mu/\mu, \lambda)$-theory, *Evolutionary Computation* **3**, 1, pp. 81–111.

Beyer, H.-G. (2001). *The Theory of Evolution Strategies*, Natural Computing Series (Springer, Heidelberg).

Beyer, H.-G. and Sendhoff, B. (2008). Covariance matrix adaptation revisited - the CMSA evolution strategy, in G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni and N. Beume (eds.), *Proceedings of PPSN*, pp. 123–132.

Conn, A., Scheinberg, K. and Toint, L. (1997). Recent progress in unconstrained nonlinear optimization without derivatives, URL `citeseer.ist.psu.edu/`

conn97recent.html.

de Boer, P.-T., Kroese, D., Mannor, S. and Rubinstein, R. (2005). A tutorial on the cross-entropy method, *Annals of Operations Research* **134**, 1, pp. 19–67, doi:10.1007/s10479-005-5724-z, URL http://dx.doi.org/10.1007/s10479-005-5724-z.

Devroye, L., Györfi, L. and Lugosi, G. (1997). *A probabilistic Theory of Pattern Recognition* (Springer).

Droste, S. (2005). Not all linear functions are equally difficult for the compact genetic algorithm, in *Proc. of the Genetic and Evolutionary Computation COnference (GECCO 2005)*, pp. 679–686.

Fournier, H. and Teytaud, O. (????). Lower bounds for comparison based evolution strategies using VC-dimension and sign patterns, , pp. 1–22.

Fournier, H. and Teytaud, O. (2008). Lower bounds for evolution strategies using VC-dimension, in G. Rudolph, T. Jansen, S. M. Lucas, C. Poloni and N. Beume (eds.), *PPSN*, *Lecture Notes in Computer Science*, Vol. 5199 (Springer), ISBN 978-3-540-87699-1, pp. 102–111.

Gelly, S., Ruette, S. and Teytaud, O. (2007). Comparison-based algorithms are robust and randomized algorithms are anytime, *Evolutionary Computation Journal (MIT Press), Special issue on bridging Theory and Practice* **15**, 4, pp. 411–434.

Hansen, N. and Ostermeier, A. (2001). Completely derandomized self-adaptation in evolution strategies, *Evolutionary Computation* **9**, 2, pp. 159–195.

Harik, G. R., Lobo, F. G. and Goldberg, D. E. (1999). The compact genetic algorithm, *IEEE Trans. on Evolutionary Computation* **3**, 4, p. 287.

Hooke, R. and Jeeves, T. A. (1961). "Direct search" solution of numerical and statistical problems, *Journal of the ACM* **8**, 2, pp. 212–229.

Kolmogorov, A.-N. and Tikhomirov, V.-M. (1961). ε-entropy and ε-capacity of sets in functional spaces, *Amer. Math. Soc. Transl. 17, pp 277-364* .

Larranaga, P. and Lozano, J. A. (2001). *Estimation of Distribution Algorithms. A New Tool for Evolutionary Computation* (Kluwer Academic Publishers).

Matoušek, J. (2002). *Lectures on Discrete Geometry*, *Graduate Texts in Mathematics*, Vol. 212 (Springer).

Mühlenbein, H. and Höns, R. (2005). The estimation of distributions and the minimum relative entropy principle, *Evolutionary Computation* **13**, 1, pp. 1–27.

Mühlenbein, H. and Mahnig, T. (2002). Evolutionary computation and Wright's equation. *Theoretical Computer Science* **287**, 1, pp. 145–165.

Nelder, J. and Mead, R. (1965). A simplex method for function minimization, *Computer Journal 7* , pp. 308–311.

Powell, M. J. D. (1974). Unconstrained minimization algorithms without computation of derivatives, *Bollettino della Unione Matematica Italiana* **9**, pp. 60–69.

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

Rónyai, L., Babai, L. and Ganapathy, M. K. (2001). On the number of zero-

patterns of a sequence of polynomials, *Journal of the American Mathematical Society* **14**, 3, pp. 717–735.

Rudolph, G. (1997). Convergence rates of evolutionary algorithms for a class of convex objective functions, *Control and Cybernetics* **26**, 3, pp. 375–390.

Sauer, N. (1972). On the density of families of sets, *Journal of Combinatorial Theory, Ser. A* **13**, 1, pp. 145–147.

Schwefel, H.-P. (1974). Adaptive Mechanismen in der biologischen Evolution und ihr Einfluss auf die Evolutionsgeschwindigkeit, Interner Bericht der Arbeitsgruppe Bionik und Evolutionstechnik am Institut für Mess- und Regelungstechnik Re 215/3, Technische Universität Berlin.

Teytaud, O. (2007). On the hardness of offline multi-objective optimization, *Evolutionary Computation* **15**, 4, pp. 475–491.

Teytaud, O. and Gelly, S. (2006). General lower bounds for evolutionary algorithms, in *Proceedings of PPSN*, *Lecture Notes in Computer Science*, Vol. 4193 (Springer), pp. 21–31.

Vapnik, V. and Chervonenkis, A. (1968). On the uniform convergence of frequencies of occurence events to their probabilities, *Soviet Mathematics-Doklady 9, 915-918* .

Vapnik, V. and Chervonenkis, A. (1971). On the uniform convergence of relative frequencies of events to their probabilities, *Theory of Probability and its Applications* **16**, 2, pp. 264–280.

Whitley, D. (1989). The GENITOR algorithm and selection pressure: Why rank-based allocation of reproductive trials is best, in J. D. Schaffer (ed.), *Proceedings of the Third International Conference on Genetic Algorithms* (Morgan Kaufman, San Mateo, CA), pp. 116–121.

Witt, C. and Jägersküpper, J. (2005). Rigorous runtime analysis of a (mu+1) es for the sphere function, in *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation* (ACM), pp. 849–856.