General lower bounds for evolutionary algorithms

Olivier Teytaud, Sylvain Gelly

General lower bounds for evolutionary algorithms

Olivier Teytaud, Sylvain Gelly

TAO (Inria), LRI, UMR 8623(CNRS - Univ. Paris-Sud),
bat 490 Univ. Paris-Sud 91405 Orsay, France, teytaud@lri.fr

Abstract.

```
@inProceedings{lbes,
        author={O. Teytaud and S. Gelly},
        title={General lower bounds for evolutionary algorithms},
  booktitle = {$10^{th}$ International Conference on Parallel Problem
               Solving from Nature (PPSN 2006), 10 pages,},
        year=2006}
```

Evolutionary optimization, among which genetic optimization, is a general framework for optimization. It is known (i) easy to use (ii) robust (iii) derivative-free (iv) unfortunately slow. Recent work [8] in particular show that the convergence rate of some widely used evolution strategies (evolutionary optimization for continuous domains) can not be faster than linear (i.e. the logarithm of the distance to the optimum can not decrease faster than linearly), and that the constant in the linear convergence (i.e. the constant C such that the distance to the optimum after n steps is upper bounded by C^n) unfortunately converges quickly to 1 as the dimension increases to ∞. We here show a very wide generalization of this result: *all comparison-based algorithms* have such a limitation. Note that our result also concerns methods like the Hooke & Jeeves algorithm, the simplex method, or any direct search method that only compares the values to previously seen values of the fitness. But it does not cover methods that use the value of the fitness (see [5] for cases in which the fitness-values are used), even if these methods do not use gradients. The former results deal with convergence with respect to the number of comparisons performed, and also include a very wide family of algorithms with respect to the number of function-evaluations. However, there is still place for faster convergence rates, for more original algorithms using the full ranking information of the population and not only selections among the population. We prove that, at least in some particular cases, using the full ranking information can improve these lower bounds, and ultimately provide superlinear convergence results.

1 Introduction

The principle of the main stream of evolutionary computation is to use only comparison between fitness values, and not the fitness values themselves. In almost

all cases, the algorithm is indeed only based on comparisons between fitnesses of elements currently in a so-called "population", that has bounded size. Many algorithms, in spite of this restriction, have been proved linear (i.e. it has been proved that the logarithm of the distance to the optimum decreases linearly); see e.g. [4, 1, 2, 10]. Some linear lower bounds also exist in various cases ([8]). In this paper, we show that :
- this kind of algorithms can at best be linear *w.r.t the number of comparisons*, with a constant $1-O(\frac{1}{d})$ as the dimension d increases to ∞, even with very easy fitness functions ;
- however, such algorithms can have slightly better constants *w.r.t the number of function evaluations* (theorem 4), for not too strange fitness-functions ; an interesting point is that this requires features that are not present in usual (μ, λ)-algorithms ;
- in some very particular cases, these non-standard algorithms can be superlinear if they use ranking-informations and not only selection (theorem 5).
The principle of the proof of the first point is as follows. In this informal introduction, we present it in the continuous case, but the proof is general. Consider an algorithm guided by comparisons. Then, after n_c comparisons, you have at most 2^{n_c} possible behaviors (possibly stochastic, but we may think of deterministic behaviors in this informal introduction). This is not so large in front of the entropy (here quantified by the packing number): to be precise within distance ϵ in $[0,1]^d$, you must be able of at least $\Omega(1/\epsilon^d)$ different answers, i.e. you need $d\log(1/\epsilon)$ bits of information to get a precision ϵ. This rough introduction shows already that n_c ensuring a precision ϵ must be at least such that $2^{n_c} = \Omega(1/\epsilon^d)$, i.e. ϵ decreases as $(2^{-1/d})^{n_c}$. The convergence is therefore at most linear, with a coefficient roughly $2^{-1/d} = 1 - O(1/d)$, hence the expected result that will be proved formally below. The reasonning also holds in the discrete case, and similar results can be derived for multi-modal optimization.

The proof of the second point, better constants *w.r.t the number of function evaluations*, is based on the information contained in rankings instead of selection. The proposed algorithm, realizing the task, is something between Nelder-Mead algorithm ([9]) and evolution strategies. The fitness is built in an ad hoc manner, but has some reasonnable properties that make the proof not too artificial. This is absolutely not the case of the proof of the last point (superlinearity w.r.t of the number of function evaluations), which uses ad hoc fitness and algorithm which are of theoretical but not practical interest. We do not know if this result also holds for more natural functions.

2 General Framework

Let D be the domain in which we look for optimal (say, minimal) values of a given real-valued function called the fitness. The packing number of a set S with respect to a metric and some $\epsilon > 0$, is the maximal number of points (possibly ∞) such that (i) each point is in S (ii) any two distinct points are at distance at least ϵ. We note $|S|$ the cardinal of the set S (possibly infinite). We note $E_{y_1,y_2,...,y_k}$ the expectation with respect to random variables $y_1, \ldots, y_k$. A

property F being given, $\mathbf{1}_F$ denotes the function with value 1 when F holds, and 0 otherwise else. Let g_n, f, h be possibly stochastic functions. See 2 for more details on assumptions. The algorithm is as follows :

1. initialize s_1 and t_1 to some value s and t.
2. for $n = 1$ to ∞, do $Epoch(n)$ which is
   (a) compute $(r_n, t_{n+1}) = g_n(s_n, t_n)$ with $r_n \in K_n$;
   (b) $s_{n+1} = f(s_n, r_n)$.
   (c) consider $x_n = h(s_n)$ as your current proposal as an approximation of the min-argument of the fitness.

The goal of this algorithm is the fast convergence of x_n to the min-argument of the fitness.

No assumption is made on s_n, it can live in any domain, the only requirement is that s_{n+1} is only a function of s_n and r_n. In natural cases s_n can be a backup of all results of comparisons and of all random generations. Similarly, t_n can live in any domain, provided that t_{n+1} only depend on s_n and t_n; a natural case for us is that t_n contains all the archive of visited points with their fitness values. Also, theorems below hold for any case of K_n whenever the natural case for this paper is that r_n is the result of one or finitely many comparisons. We will now see how evolutionary algorithms fit in this framework.

Why this algorithm includes evolutionary computation. Our framework is very general, but we show here why this framework is relevant for all forms of evolutionary computation. Typically, in evolutionary computation:
- t_n is the archive of visited points with their fitness values
- r_n is the result of various comparisons (in g_n, fitness are computed and compared to other points in the population or in the archive);
- f is the method for creating generations (which might include cross-over, mutations, selection,...). Without loss of generality, we have assumed that f does not depend on n; if we want f depending on n, we just have to add the information "n" in s_n (i.e., replace s_n by (s_n, n)).

What are our hypothesis. The algorithm depends on (i) the initialization of s_1 and t_1 ; (ii) the possibly stochastic function g_n ; (iii) the possibly stochastic function f ; (iv) the possibly stochastic function h. The g_n are the only functions that use the fitness to be optimized. K_n (in which r_n has its values) is finite.

Typically, the function to be optimized is used only as a black-box. However, we will not have to assume this here in the main results. We mainly need the fact that the information provided by the fitness fits in a finite number of bits (i.e. finiteness of K_i). This is why algorithms like BFGS or even the gradient descent are not concerned by our results ; the information provided by a gradient does not fit in a finite number of bits (at least if we assume infinite precision of floating-point values). The t_n can have any structure, it may contain an archive of fitness values or not. We don't even need the fact that the computation time per epoch is bounded.

These assumptions are typically true for evolutionary algorithms. As we do not assume anything about the structure of s_n, and we do not assume anything

about what is done with the fitness except that the number of bits is small, our result is much more general than only evolutionary computation.

3 Lower bounds w.r.t of the number of comparisons

With hypotheses of 2, we claim :

Entropy Lemma for Evolutionary Computation:

Consider a set Fit of possible fitness functions on domain D, i.e. $Fit \subset \mathbb{R}^D$, such that any $fit \in Fit$ has only one min-argument fit^*, and such that $\{fit^*; fit \in Fit\} = D$. This means that we don't know a priori where is the min-argument (Fit can be the set of sphere functions or any other very simple optimization problems).

Then, define $N(\epsilon)$ the packing number of D for 2ϵ, for some metric. Consider fit a random variable such that fit^* is uniformly distributed on the $N(\epsilon)$ elements realizing the packing number of D. Consider some fixed $\delta \in]0,1[$ and n such that the probability (on both fit and x_n) that $d(x_n, fit^*) \leq \epsilon$ (where $d(.,.)$ is the euclidean distance) verifies $P(d(x_n, fit^*) \leq \epsilon) \geq 1 - \delta$.

Then $n \geq n_{\epsilon,\delta} = \lceil \frac{\log(1-\delta)}{\log(K'_n)} + \frac{\log(N(\epsilon))}{\log(K'_n|)} \rceil$ where $K'_n = \sqrt[n]{\prod_{i=1}^n |K_i|}$.

Proof: We note S_ϵ the set of points of the domain that lie at distance $\leq \epsilon$ of the optimum fit^* for the $||.||_\infty$ norm.

Step 1: conditionning to a sequence of r_i. Consider a fixed sequence of (r_i), instead of r_i function (via g_i) of s_i and t_i. We consider a run of the algorithm, in the case in which these r_i are fixed. Then, conditionally to x_n, $E_{fit}\mathbf{1}_{\{x_n \in S_\epsilon\}} \leq \frac{1}{N(\epsilon)}$. Averaging on x_n (which can be random if the algorithm is stochastic) leads to $E_{fit}E_{x_n}\mathbf{1}_{\{x_n \in S_\epsilon\}} \leq 1/N(\epsilon)$.

Step 2: summing on all possible $(r_i)_{i\in[[1,n]]}$

$$E_{fit} \sup_{r_n} E_{x_n}\mathbf{1}_{\{x_n \in S_\epsilon\}} \leq E_{fit} \sum_{r_n} E_{x_n}\mathbf{1}_{\{x_n \in S_\epsilon\}}$$

$$\leq \sum_{r_n} E_{fit}E_{x_n}\mathbf{1}_{\{x_n \in S_\epsilon\}} \leq \prod_{i=1}^n \frac{|K_i|}{N(\epsilon)} \text{ thanks to step 1.}$$

Note for short $K'_n = \sqrt[n]{\prod_{i=1}^n |K_i|}$. Then, $P(d(x_n, fit^*) \leq \epsilon) \geq 1 - \delta$ implies ${K'_n}^n/N(\epsilon) \geq 1 - \delta$. This implies that $n \geq \log(1-\delta)/\log(K'_n) + \log(N(\epsilon))/\log(K'_n)$. □

Note that for many algorithms, K_n is constant, and therefore K'_n is constant. An easier formulation is as follows :

Theorem 1: Entropy Theorem for Evolutionary Computation:

Consider a set Fit of possible fitness functions on domain D, i.e. $Fit \subset \mathbb{R}^D$, such that any $fit \in Fit$ has only one min-argument fit^*, and such that $\{fit^*; fit \in Fit\} = D$. This means that we don't know a priori where is the min-argument (Fit can be the set of sphere functions or any other very simple optimization problems). Consider some fixed $\delta \in]0,1[$ and n such that for any fitness in Fit, $P(d(x_n, fit^*) \leq \epsilon) \geq 1 - \delta$. Then $n \geq n_{\epsilon,\delta} = \lceil \frac{\log(1-\delta)}{log(K'_n)} + \frac{\log(N(\epsilon))}{\log(K'_n)} \rceil$ where $K'_n = \sqrt[n]{\prod_{i=1}^n K_i}$.

Proof: Assume, to get a contradiction, that n ensures $d(x_n, fit^*) \leq \epsilon$ with probability $1-\delta$ for any fitness in Fit. Then, *a fortiori*, it ensures it with probability $1-\delta$ on average for any distribution of fitnesses in Fit. This leads to a contradiction with the previous lemma, hence the expected result.□

This theorem provides the convergence rate with respect to the number of epochs. This is in particular interesting when (i) each epoch is parallelized ; or (ii) the cost is mainly the cost of fitness-evaluations and the number of fitness evaluations per epoch is some fixed q fitness-evaluations/epoch, what implies that the average coefficient of linear convergence per fitness evaluation r_{fe} is $r_{fe} = \sqrt[q]{r_e}$ where r_e is the coefficent of the linear convergence per epoch.

These lower bounds are absolute lowers bounds with respect to the epochs, but we might also be interested in bounds with respect to time, without neglecting the computational cost of each epoch. We can do this with a very natural assumption, very classical in evolutionary algorithms, which is that we only use comparisons on the fitness values. We can then derive a bound on the convergence rate with respect to the number of comparisons, and therefore on the convergence rate with respect to time :

Corollary 2 (entropy theorem for black-box evolutionary algorithms: complexity w.r.t. number of comparisons): *Assume the same hypothesis as in the theorem above with $K_n = 2$ corresponding to r_n equal to the result of a comparison between the fitnesses of two previously visited points. 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 $n_c \geq \log_2(1-\delta) + \log_2(N(\epsilon))$. I.e., formally, $P(||x_{n_c} - fit^*|| < \epsilon) \geq 1-\delta \Rightarrow n_c \geq \log_2(1-\delta) + \log_2(N(\epsilon))$.*

Proof: Split the algorithm in section 2 so that each epoch contains at most one comparison. Then $|K'_n| = |K_n| = 2$ as any computation except the comparison can be put in f. Hence the expected result.□

Corollary 2': the same with respect to area. *If the domain has measure 1, and if Fit has the same property as in theorem 1, then n_c comparisons are necessary for a comparison-based algorithm in order to provide a set with measure $v < 1$ that contains fit^* with probability at least $1-\delta$, where $n_c \geq \log_2(1-\delta) + \log_2(1/v)$.* and also with notations as above, the number of epochs n verifies $n \geq \log(1-\delta)/\log(K'_n) + \log(1/v)/\log(K'_n)$.

Proof: The proof is very similar to the previous one, and is indeed simpler. Consider a fixed sequence of r_n. Consider fit a random variable on Fit such that fit^* is uniform on the domain D. Note V the set proposed by the algorithm, and that must contain fit^* with probability at least $1-\delta$.

Consider a fixed V. Then, the probability (on fit) that $fit^* \in V$ is at most v. Now, by averaging on V (conditionaly to a sequence of r_n), we have $P_V(fit^* \in V) \leq v$. If we now consider the sum of these probabilities among possible sequences of r_n, we have $P(fit^* \in V) \leq 2^{n_c} v$ and therefore $1-\delta \leq 2^{n_c} v$, which leads to $n_c \geq \log_2(1-\delta) - \log_2(v)$ where $\log_2(t) = \log(t)/\log(2)$.□

Continuous case: linear convergence w.r.t the number of comparisons. The bound above on the convergence rate depends on the packing number of the domain. This bound holds for any families of fitnesses, provided that the

optimum is not known in advance (i.e. it can be anywhere in the domain). We will now apply it to the simple continuous case $D = [0,1]^d$ with the supremum norm, $N(\epsilon) \geq (\lceil 1/\epsilon \rceil^d)$. First consider the convergence with respect to n the number of epochs in the (standard) case: $\forall i, K_i \leq K$ for some K. This implies that the guaranteed distance to the optimum, for some n and with probability at least $1-\delta$, for fixed δ, verifies $N(\epsilon) \leq K^n/(1-\delta)$ i.e. $\lceil 1/\epsilon \rceil \leq (K^n/(1-\delta))^{1/d}$, i.e. $\epsilon \geq 1/\left(1+(K^n/(1-\delta))^{1/d}\right)$. This is (at best) a linear convergence, with constant in $[1-O(1/d), 1]$. The convergence with respect to time if only comparisons are used is more strongly bounded, as shown in the corollary (without assuming anything except the fact that only comparisons of fitnesses are used) : $\epsilon \geq 1/\left(1+(2^{n_c}/(1-\delta))^{1/d}\right)$ where n_c is the number of comparisons. This is (at best) a linear convergence, with constant in $[1-O(1/d), 1]$, independent of the algorithm for a fixed K. We will see below that modifying K for example by modifying λ and μ does not significantly modify the result w.r.t the number of comparisons, but it does w.r.t the number of function-evaluations, *but only if we use full-ranking and not only selection*. Note that the bound is tight: the following problems $\{x \mapsto ||x - fit^*||_1; fit^* \in [0,1]^d\}$ is solved with constant $1 - \Omega(1/d)$ by the following algorithm (close to the Hooke&Jeeves algorithm [7]), that reaches $1 - \Theta(1/d)$ both w.r.t the number of fitness-evaluations and w.r.t the number of comparisons:

- initialize $x = (x_1, \ldots, x_d)$ at any point.
- in lexicographic order on $(j,i) \in \mathbb{N} \times [[0, d-1[[$:
  - try to replace the j^{th} bit b of x_i by $1-b$;
  - if it is better, then keep the new value; otherwise else keep the old value.

4 Convergence rate with respect to the number of fitness-evaluations: why the $1 - O(1/d)$ is also true for selection-based algorithms

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 rate will be $\geq 1/\sqrt[\lambda d]{K}$.

- (μ, λ)**-ES (or SA-ES) :** at each step, then, we only know which are the selected points. Then, $K = \binom{\lambda}{\mu} \leq \binom{\lambda}{\lfloor \lambda/2 \rfloor} \leq (2^\lambda/\sqrt{2\pi\lambda})$ (see e.g. [3, p587] or [6] for proofs about $\binom{\lambda}{\mu}$). This leads to a convergence rate with respect to the number of FEs $> 1/\sqrt[\lambda d]{2^\lambda} \geq 1/\sqrt[d]{2}$, hence the $1 - O(1/d)$ result with respect to the number of FEs ; note that the constant is *worst* if λ increases.

- Consider **more generally, any selection based algorithm**, i.e. any algorithm in which r_n encodes only a subset of μ points among λ. Then, the algorithm provides only a subset of $[[1, \lambda]]$, i.e. $K_n \leq 2^\lambda$, and $\sqrt[\lambda]{K} = O(1)$ and the convergence rate is $\geq 1 - O(1/d)$ bound in distance or $\exp(-O(1))$ in area. Note that this remains true if λ depends on the epoch and even if the subset has not a fixed size defined before the fitness-evaluations. As we will show below,

this $1 - O(1/d)$ with respect to the number of FEs **is not true for algorithms using the full ranking information** ; this allows the conclusion that **using *all* the ranking information can lead to better convergence rates, at least in some cases, than using *only* a *selection* information**.

- $(\mu + \lambda)$**-ES (or SA-ES) :** then, we only know which are the selected points. Then, $K = (\lambda + \mu)!/(\mu!\lambda!)$. This does not allow a proof of $1 - O(1/d)$ if μ increases as a function of d, but indeed, for $(\mu + \lambda)$-ES, the $1 - O(1/d)$ can be proved by other means ; see e.g. [11]. Note however that for other algorithms (not $(\mu + \lambda)$-ES) with a big archive (what is somewhat similar to a big μ), we will see that the $1 - O(1/d)$ does not hold.

- **Parallel** $(1 + \lambda)$**-ES:** As the λ points are computed in parallel, we don't need to consider the $\sqrt[\lambda]{(.)}$; the convergence rate is $\geq 1/\sqrt[d]{K} = 1/\sqrt[d]{\lambda}$. Here, $K \leq \lambda$, therefore the speed up is only at most logarithmic (the number of fitness-evaluations required for a given precision decreases only as $\log(N(\epsilon))/\log(\lambda)$).

Consider the convergence rate with respect to the area as in corollary 2', with respect to epochs. For an averaged convergence rate with respect to the number of fitness-evaluations, we must consider the λ^{th} root ; the convergence rate in area is $O(\sqrt[\lambda]{\frac{2^\lambda}{\sqrt{\lambda}}})$. Increasing the population size to infinity as dimension increases will therefore not improve the result in (μ, λ) schemas: this leads to $\exp(-o(1))$ instead of the $\exp(-\Theta(1))$ that can be reached by some algorithm like (1+1)-ES. Therefore, in the case of (μ, λ)-ES, either the population remains finite, and we can reach $\exp(-\Theta(1))$, or the population increases to infinity as the dimension increases and it is worse.

The case of $(\mu + \lambda)$-ES is different, as a huge μ has no cost w.r.t function evaluations (is only involves archiving). With a huge μ, as we have no restriction here on the selection method and the information stocked in memory and the computational power (we only count the number of fitness-evaluations), you can encode in an archive many specific methods, and in particular the algorithms below beating the $\exp(-O(1))$ (for area with respect to convergence rates). However, note that for standard $(\mu + \lambda)$-ES, the numerical evaluation of the bounds above, which depends on the rule for specifying μ and λ as functions of the dimension, lead to $\exp(-O(1))$ at best (for the area, with respect to the number of function evaluations).

5 Superlinearity: what about the complexity with respect to the number of fitness-evaluations ?

K_n can run to infinity as $n \to \infty$. This implies that the computational cost of an epoch converges to infinity, but this might happen in particular if the principal cost is the evaluation of the fitness. For example in algorithms using the full archive of visited points and using the ranking of all visited points, we can compare each new point to all previously visited points. Can this improve the result in term of convergence rate with respect to the number of visited points? For the moment, we have bounds with respect to the computational time (corollary above), with respect to the number of epochs (the main theorem), but what about a bound on the convergence rate with respect to the number of

fitness-evaluations, neglecting the other computational costs ? Such bounds are important as evolutionary algorithms are particularly relevant for very expensive fitnesses. Section 4 answers partially to this question for some algorithms. A positive result is a clue for designing algorithms that might be superlinear, or might have better dependencies on the dimension. We will show below that using full ranking information, it is possible to outperform the $1 - O(1/d)$ that hold, even w.r.t the number of function-evaluations for selection based algorithms.

The ultimate limit of corollary 2 w.r.t function-evaluations. Assume that we only use comparisons (but allow as many comparisons as you want per epoch). Then, let's rewrite the algorithm so that there is only one call to the fitness function per epoch. This only means that we split each epoch in the number of fitness-evaluations. Then, we see that there are at most n possible outcomes in the set of comparisons in this epoch: the rank of the newly evaluated point. This implies that $K_n \leq n$. Then, the number of epochs required to ensure a precision ϵ with probability $1 - \delta$ is $n \geq \log(1-\delta)/\log(K'_n) + \log(N(\epsilon))/\log(K'_n)$ with $K'_n = \sqrt[n]{n!} = \Theta(n)$. In the continuous case, this is asymptotically (slightly) superlinear, but at the cost of a computation time per epoch increasing to infinity. Let's summarize these elements.

Corollary 3: convergence rate w.r.t. the number of fitness-evaluations. *Assume that K_n contains only the result of comparisons between values of the fitness at visited points. Then, the number of visited points necessary for a precision at most ϵ with probability at least $1 - \delta$ is at least $n_{fe} \geq \log(1-\delta)/\log(K'_n) + \log(N(\epsilon))/\log(K'_n)$ with $K'_n = \Theta(n)$ (i.e. a superlinear convergence rate in the continuous case $[0,1]^d$).*

Whereas (as shown in corollary 2) the number of comparisons required is at least $n_c \geq \log_2(1-\delta) + \log_2(N(\epsilon))$ (i.e. a linear convergence rate in the continuous case $[0,1]^d$, with coefficient $1 - O(1/d)$).

This suggests the possible relevance of evolutionary algorithms for expensive fitnesses, for which the computational cost of each epoch out ot fitness-calls is negligible: for low-cost fitness, where the computational cost of the comparisons is not negligible, we know that we can not be superlinear, and that the constant quickly runs to 1 as the dimension increases, but we let open the possibility of superlinearity w.r.t the number of fitness evaluations, and the possibility of constants better than this $1 - O(1/d)$.

In particular, our proof above (corollary 2') forbids better than $\exp(-O(1))$ in the following terms: *if the domain has measure 1, then the number of comparisons required by a comparison-based algorithm for providing an area of measure $v < 1$ that contains the optimum with probability at least $1 - \delta$ is as least* $n_c \geq \log_2(1-\delta) + \log_2(1/v)$. This is a bound in $\exp(-O(1))$ for the convergence rate with respect to the area, uniformly in all the possible dimensions. It is in some sense more natural, because it reflects the idea that in order to divide the area where the optimum can lie by 2, you need 1 bit of information. This bound
- holds with respect to the number of comparisons (corollary 2') ;
- holds with respect to the number of fitness-evaluations if the number of comparisons per epoch is a priori bounded independendly of the dimension or under

various hypothesis including nearly all existing comparison-based algorithms ;
- but does not hold w.r.t the number of fitness-evaluations in the general case. Indeed, it is possible to avoid the $\exp(-O(1))$ if the population size increases to infinity, on some not too artificial fitness-functions. If we look to very particular cases of fitness-functions, it is also possible to be superlinear w.r.t the number of fitness-evaluations, with only comparisons. This point will be shown below.

Improved convergence rates using full ranking information. We now formalize two theorems about this precise point. The first one considers the convergence better than $\exp(-O(1))$ from the area point of view on a reasonnable fitness, thanks to the use of a bigger information than only the selected points: the algorithm uses the full ranking of the population. The second one reaches superlinearity, but for a very particular fitness and a very particular algorithm, so is only of theoretical interest.

Theorem 4: better than $\exp(-O(1))$ for the convergence rate of the area. *In spite of the various results showing bounds in $\exp(-O(1))$ on the constant in linear convergence rates, it is possible under the following hypotheses:*
- continuous domain with non-empty interior and dimension d;
- family of fitnesses that satisfy the hypothesis of theorem 1 (for any $fit^* \in D$, there is at least one fitness fit with optimum in fit^*) ;
- fitnesses radially increasing ($\forall x \neq 0, t > 0, t \mapsto fit(fit^* + tx)$ is increasing);
- comparison-based algorithm;
to reach a $O(1/d)$ constant from the point of view of the convergence of the area with respect to the number of function-evaluations.

Remark: Selection is strictly less informative than ranks. Theorem 4 shows that it is possible to outperform the $\exp(-O(1))$ in area in a framework using only ranks. We have shown above that algorithms based on selections only could not outperform $\exp(-O(1))$. Therefore, at least for particular fitness-functions, full ranking is significantly more informative than selection only (i.e., can lead to $o(1)$ instead of $\exp(-O(1))$). In the same spirit, theorem 5 (superlinearity) can not be reached with selection only.

Proof: Let S be a regular simplex in $\mathbb{R}^d$ with diameter $\frac{1}{2}$ and vertices $v_0, \ldots, v_d$. Consider $S_0, \ldots, S_{k-1}$ the k simplices corresponding to the $k = (d+1)!$ permutations $\sigma_0, \ldots, \sigma_{(d+1)!}$ of $[[0, d]]$ as follows: $S_i = \{x \in S; \forall j \in [[0, d-1]]\ d(x, v_{\sigma(j)}) \leq d(x, v_{\sigma(j+1)})\}$. S is partitioned in the S_i, within frontiers which have null measure. Define $\delta(x)$ such that $x \in S_{\delta(x)}$ for almost all x in S. Then, define π_i a linear one-to-one mapping from S_i to S. Define for $x \in S$, $\pi(x) = \pi_{\delta(x)}$. Note $\pi^i(x) = \pi(\pi^{i-1}(x))$ with π^0 the identity. Define the fitness fit associated to an optimum fit^* as follows : $fit(x) = -1 - i + d(\pi^i(x), \pi^i(fit^*))$ where i is minimal such that $\delta(\pi^i(x)) \neq \delta(\pi^i(fit^*))$ and $fit(x) = -\infty$ is no such i exists. This fitness has unbounded negative values, but one can consider $-1/fit(x)$ positive fitnesses are prefered. Consider the following algorithm :

- initialize at $T_0 = S$ and f_0 at the identity from S to $T_0 = S$.
- for $n = 0, \ldots, \infty$:
  - evaluate the $d+1$ vertices $w_0 = f_n(v_0), \ldots, w_d = f_n(v_d)$ of T_n ;

- set $T_{n+1} = f_n(S_i)$ where i is such that $S_i = \{x; (d(x, v_j))_{j\in[[0,d]]}$ is in the same order as $(fit(w_j))_{j\in[[0,d]]}\}$
- set f_{n+1} a linear one-to-one mapping from S to T_{n+1}.

On the fitness defined above, the algorithm verifies for all $n \in \mathbb{N}$: (i) $fit^* \in T_n$; (ii) the area of T_n divided by the area of T_0 is $\left(\frac{1}{(d+1)!}\right)^n$. The area therefore decreases of $(d+1)!$ per epoch ; each epoch contains $d+1$ evaluations. The average convergence rate for the area is therefore $\sqrt[d+1]{\frac{1}{(d+1)!}} = \Theta(\frac{1}{d})$.□

Theorem 5: superlinear convergence rates w.r.t. number of function evaluations. *There is one algorithm and one family of fitness functions such that (i) for almost all fit^* in the domain D there is a fitness fit with only one optimum at fit^* ; (ii) the convergence is superlinear.*

Proof:

Define $P_n(D)$ the set of subsets of D with cardinal n.

For any measurable subset T of the domain D, define $p_n(T)$ as a partition of T in $n!$ measurable sets of equal measure. We define by induction

- $\mathcal{T}_1 = \{D\}$;
- $\mathcal{T}_{n+1}$ the minimal set of subsets of D such that $T \in \mathcal{T}_n \Rightarrow p_n(T) \subset \mathcal{T}_{n+1}$.

We note $p_{n,i}(T)$ the i^{th} element of $p_n(T)$ (for any arbitrary order). We note $(\sigma_{k,n})_{k\in[[1,n!]]}$ an enumeration of the permutations of $[[0, n-1]]$. Let h_n be a deterministic function from $\mathcal{T}_n$ to $P_n(D)$, which associate to $T \in \mathcal{T}_n$, n points on the frontier of T. We assume that $h_n(T) \cap h_m(T') = \emptyset$ for any $T \neq T'$ in $\mathcal{T}_n \times \mathcal{T}_m$.

The algorithm is as follows :

- $T_1 = D$;
- for $n = 1$ to infinity,
  - choose deterministically, depending on T_n only, n points on the frontier of T_n to be evaluated: $\{w_0, \dots, w_{n-1}\} = h_n(T_n)$;
  - let k be such that $fit(w_{\sigma_{k,n}(0)}) \leq fit(w_{\sigma_{k,n}(1)}) \leq \dots \leq fit(w_{\sigma_{k,n}(n-1)})$;
  - define $T_{n+1} = p_{n,k}(T_n)$.

Then, the fitness with optimum at fit^* on which this algorithm converges superlinearly, is defined as follows :

- if $x \in \cup_n \{h_n(T); T \in T_n\}$, then $fit(x)$ is equal to $\sigma_{k,n}(i) - n!$ where
  - $fit^* \in T$ with $T \in \mathcal{T}_n$,
  - $w_i = x$ and $w = h_n(T)$,
  - k is such that $fit^* \in p_{n,k}(T)$;
- otherwise else, for any $x \neq fit^*$, $fit(x) = ||x - fit^*||^2$, and $fit(fit^*) = -\infty$.

As in theorem 4, one can replace this fitness with minimum $-\infty$ by an equivalent bounded fitness (e.g. $1/\exp(-fit)$).

The proof is now straightforward :

- by induction (and construction) T_n contains fit^* for any n ;

– the area of T_n divided by the area of T_{n+1} is $n!$;
– the average convergence rate is $1/\sqrt[n]{n!} = \Theta(1/n)$ that runs to 0 as $n \to \infty$. □

This proof is very artificial (the best T_{n+1} is encoded in the ranking of fitness values at the points in $h(T_n)$) but has some advantages :

– the convergence rate is superlinear, what shows that it is possible, at least for very particular cases ;
– the convergence rate for area could be translated to convergence rate for distance, if we take care of partitionning in a way such that the diameter of T_n is linear in the area of T_n (leading to a superlinear convergence rate for the distance to optima also).

6 Conclusion

We have studied algorithms that only depend on comparisons. We have shown (section 3) that ranking-based methods can not be better than linear, and that the constant runs to 1 as the dimension d runs to infinity, at least as $1 - O(1/d)$. The result does not only concern comparison-based methods, it concerns all algorithms using at each epoch finitely many bits of information (what is not the case of algorithm using real numbers, at least on ideal computers). This linearity and this constant are with respect to the number of bits, e.g. the number of comparisons. In section 4, similar results are derived for the convergence with respect to the number of function evaluations. We also show that increasing λ e.g. as dimension increases does not improve the result, in a stronger sense for (λ, μ) algorithms than for $(\lambda + \mu)$-algorithms. However, these negative results, that generalize the state of the art, does not formally forbid superlinearity for comparison-based algorithms w.r.t the number of fitness-evaluations. We have then (section 5) shown that superlinearity w.r.t the number of function evaluations is possible. The contrast with the results of section 3 show that superlinear algorithms can only be superlinear w.r.t the number of function-evaluations (and not the number of comparisons), and that traditional (λ, μ)-ES or SAES or any usual algorithm can't be superlinear. Superlinear algorithms, or even linear algorithms with better constants as d increases, must use a stronger information from comparisons, typically the full ranking, and not only selection. We have exhibited such algorithms, one of them which is reasonnable (theorem 4, improving the dependency in front of the dimension by a Nelder-Mead inspired algorithm, modified for taking into account the full ranking) and one of them purely theoretical (theorem 5, superlinearity).

Acknowledgements

Many thanks to Anne Auger; talks with her initiated this work through the idea of the class of comparison-based algorithms. Thanks also to Jens Jägerskϋpper, Stefan Droste, Thomas Jansen, Marc Schoenauer and various people from Dagsthul's seminar "theory of evolutionary algorithms 2006" for many interesting talks. This work was also supported in part by the Pascal Network of Excellence.

References

1. A. Auger. Convergence results for (1,λ)-SA-ES using the theory of φ-irreducible markov chains. *Theoretical Computer Science*, 2005. in press.
2. A. Auger, M. Jebalia, and O. Teytaud. Xse: quasi-random mutations for evolution strategies. In *Proceedings of Evolutionary Algorithms, 12 pages*, 2005.
3. L. Devroye, L. Györfi, and G. Lugosi. *A probabilistic Theory of Pattern Recognition*. Springer, 1997.
4. S. Droste. Not all linear functions are equally difficult for the compact genetic algorithm. In *Proc. of the Genetic and Evolutionary Computation COnference (GECCO 2005)*, pages 679–686, 2005.
5. S. Droste, T. Jansen, and I. Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization, 2003.
6. W. Feller. *An introduction to Probability Theory and its Applications*. Willey, 1968.
7. R. Hooke and T. A. Jeeves. Direct search solution of numerical and statistical problems. *Journal of the ACM, Vol. 8, pp. 212-229*, 1961.
8. J. Jagerskupper and C. Witt. Runtime analysis of a (mu+1)es for the sphere function. Technical report, 2005.
9. J. Nelder and R. Mead. A simplex method for function minimization. *Computer Journal 7*, pages 308–311, 1965.
10. G. Rudolph. Convergence rates of evolutionary algorithms for a class of convex objective functions. *Control and Cybernetics*, 26(3):375–390, 1997.
11. O. Teytaud, S. Gelly, and J. Mary. On the ultimate convergence rates for isotropic algorithms and the best choices among various forms of isotropy, ppsn 2006.