The Curse of High-Dimensional Search Spaces: Observing Premature Convergence in Unimodal Functions

Ralf Salomon
Department of Electrical Engineering and Information Technology
University of Rostock, 18051 Rostock, Germany
Email: ralf.salomon@etechnik.uni-rostock.de

***Abstract*— It might come to a surprise, if some properly designed evolutionary algorithms might not be able to sufficiently optimize simple, continuous, unimodal objective functions. Keeping in mind the evolutionary algorithms' high performance especially in the field of global function optimization, getting stuck (also known as premature convergence) at suboptimal function values is attributed mostly to the presence of distracting local optima. This paper describes some examples in which such a behavior occurs. It also gives some explanations of the underlying reasons. These explanations indicate, as a conclusion, that premature convergence might happen very well at continuous, unimodal functions (and consequently in real-world applications).**

I. Introduction

No doubt, evolutionary algorithms have advanced as mature optimization methods. As a serious alternative to rather classical (gradient-based) optimization procedures, they are being applied in diverse areas, such as financial forecasting, evolutionary robotics, VLSI design, global function optimization, operations research, and scheduling. As a prerequisite for later explanations, Section II presents the outline of evolutionary algorithms as well as some theoretical convergence considerations as far as necessary for the understanding of this paper.

A closer look to the pertinent literature [11], [9] shows that numerous classical optimization methods, such as steepest descent and Newton's method, locate the optimum $\vec{x}_o$ of a given n-dimensional unimodal objective function[1] $f(\vec{x}) = f(x_1, \ldots, x_n)$ very fast and accurately. But what happens, if those classical methods are applied to multimodal functions? Normally, they get stuck at suboptimal values. This behavior is generally attributed to the presence of (very many) distracting optima to which the procedure might be converging prematurely. This phenomenon can be observed universally, especially in interesting real-world problems.

It might be interesting to note that in his PhD thesis [3], Keneth De Jong observed that genetic algorithms, a particular class of evolutionary algorithms, are able to locate the global optimum of multimodal functions reliably under various conditions. It is widely believed that this is due to the multi-point search strategy, with which the algorithms sample many search points in parallel. A significant amount of the pertinent literature [2], [4], [7], [8], [13] provides in-depth descriptions and analyses of this quite outstanding global search capability.

Since then, the research community has devised numerous multimodal test functions for evaluation purposes (probably because unimodal test functions seem too easy). Refs. [10], [4], [13] have indicated a computational complexity of $O(n \ln n)$ for some algorithms when applied to multimodal test functions with exponentially many local optima.

It seems natural that even evolutionary algorithms cannot optimize *all* multimodal functions *equally well* (see also the discussion on the no-free-lunch theorem [16]). Anyhow, premature convergence at non-optimal fitness values is generally attributed to the existence of distracting local optima. Conversely, it would come to a surprise to many, if evolutionary algorithms would get stuck in a simple unimodal function[2]; this would way be counter-intuitive. However, Section III reports exactly this behavior. It shows that some evolution strategies, another class of evolutionary algorithms, consistently get stuck at suboptimal values in unimodal functions; these functions are continuous and do not contain any steps or plateaus (areas with constant function value).

Section IV provides some explanations for this observable behavior. In so doing, it resorts to two contradicting terms, called *gain* and *loss*. The results are then taken by Section V in order to provide some more general hints for what type of unimodal functions, evolutionary algorithms might be prone to premature convergence at suboptimal fitness values. Section VI concludes with a brief discussion.

II. Background: Algorithms and Convergence

This section summarizes some background material as far as necessary for the understanding of this paper. This includes the main concepts of evolutionary algorithms as well as a probabilistic interpretation of Beyer's progress formula [2] on quadratic functions.

A. Algorithms

1) Basic Concepts: The term evolutionary algorithms refers to a class of heuristic population-based search procedures

[1]Unimodal functions contain only one distinct optimum, whereas multimodal functions generally contain more than one.

[2]Assume for a moment that in these experiments, any *coding* or other technical problem can be and has been excluded.

that incorporate random variation and selection, and provide a framework that mainly consists of genetic algorithms [7], evolutionary programming [6], [5], and evolution strategies [12], [15].

Even though each evolutionary algorithm has its own peculiarities, they share many features. The canonical form maintains a population of μ individuals, also called parents. In each generation g, an evolutionary algorithm generates λ offspring by copying randomly selected parents and applying variation operators, such as mutation and recombination. It then assigns a fitness value (defined by a fitness or objective function) to each offspring. Depending on their fitness, each offspring is given a specific survival probability. For a good overview of these algorithms, the interested reader is referred to [1].

2) Evolution strategies and step size adaptation: Traditionally, evolution strategies represent all n parameters $x_1, \ldots, x_n$ as floating-point numbers and apply mutations by adding small random numbers. In their simplest form, evolution strategies maintain an individual-specific global step size σ, and they typically apply mutations to all n parameters, i.e., mutation probability $p_m = 1$, as follows

$$x_i \leftarrow x_i + \sigma N(0,1) \ , \tag{1}$$

with $N(0,1)$ denoting normally distributed random numbers with expectation value 0 and standard deviation 1. Each offspring inherits the step size from its parent, and prior to mutation, the inherited step size is modified by lognormally distributed random numbers[3] $\exp(N(0,1))$. This simple evolution strategy is denoted as (μ,λ)-ES or (μ+λ)-ES for short; the first notation indicates that the new parents are selected only from the offspring (i.e., no elitism), whereas the latter also considers *all* parents from the previous generation (i.e., μ-fold elitism). In addition, some evolution strategies also feature various recombination operators, such as discrete and intermediate recombination [1]. It should furthermore be mentioned that evolution strategies also feature more complex adaptation mechanisms, such as individual step sizes σ_i (one σ_i per variable x_i) as well as correlated mutations $M\vec{Z}$, with M denoting $n \times n$ correlation matrix and $\vec{Z}$ denoting an n-dimensional vector with $N(0,1)$ normally distributed, independent components z_i.

3) Two properties: The two following properties of the simple evolution strategies as described above are important for the remainder of this paper. First, the application of normally distributed random numbers to *all* parameters, which are all floating-point variables, makes the procedure rotationally invariant, i.e., independent of the actual orientation of the chosen coordinate system. In other words, the procedure's behavior is not affected by applying a pure rotation matrix Z to the chosen coordinate system $\vec{x}^* = Z\vec{x}$.

Most genetic algorithms, by contrast, do not have this property, since they heavily prefer mutations (as well as recombinations) along the coordinate system axes. This significantly influences the algorithms' analysis as well as their performance; the latter largely depends on the alignment of both the chosen parameters (i.e., the x_i's) and the problem's inherent (hidden) coordinate system.

Second, when drawing a sufficiently large number n of $N(0,1)$ normally distrubuted random numbers z_i, the resulting n-dimensional vector $\vec{Z} = (z_1, \ldots, z_n)^T$ has approximate length $\|\vec{Z}\| = \sqrt{n}$. That means, the distance d of an offspring to its parent using step size σ is approximately $d = \sigma\sqrt{n}$. Furthermore, a random vector $\vec{Z}$ with normally distributed components z_i has no preferred orientation. That means that all offspring of a particular parent are equally distrubuted on an n-dimensional hypersphere with radius $\sigma\sqrt{n}$.

B. Progress on Quadratic Functions

It is reasonable to define the rate of progress

$$\varphi = f(\vec{x}_g) - f(\vec{x}_{g+1}) \tag{2}$$

in terms of the best population members' objective function values in two subsequent generations g and g+1. For quadratic functions $f(x_1, \ldots, x_n) = \sum_i x_i^2$, [2] has derived the following rate of progress φ:

$$\varphi \approx 2Rc_{\mu\pm\lambda}\sigma - n\sigma^2 \ , \tag{3}$$

with $R = \|\vec{x}_g\|$ denoting the distance of the best population member to the optimum and $c_{\mu\pm\lambda}$ denoting a constant that subsumes all influences of the population configuration as well as the chosen selection scheme. Typical values are: $c_{1,6}$=1.27, $c_{1,10}$=1.54, $c_{1,100}$=2.51, and $c_{1,1000}$=3.24. Beyer calls the two terms on the right-hand-side *gain* and *loss*, respectively, and also considers normalized quantities:

$$\begin{aligned} \varphi^* = 2\sigma^* c_{\mu\pm\lambda} - n(\sigma^*)^2 \qquad & \text{with} \\ \varphi^* = \varphi/R^2 = \varphi/f(R), \sigma^* = \sigma/R \quad & . \end{aligned} \tag{4}$$

The remainder of this section presents an alternative derivation/interpretation of both the gain and the loss terms of the above formula. This derivation is rather probabilistic than exact. But this simplified approach can be used, to a certain extent, in other than quadratic test cases. For an excellent formal presentation, the interested reader is referred to the literature [2].

The left-hand-side of Figure 1 shows the (original) coordinate system $\vec{x}^*$ in which the currently best population member has distance R to the optimum and thus a fitness of $f(x_1^*, \ldots, x_n^*) = R^2$. Due to the rotationally invariance property of the operators (see above) and since the objective function is rotationally invariant, the coordinate system can be rotated, such that $x_1 = R$ and $x_{2..n} = 0$ holds, as the right-hand-side of Figure 1 shows. The objective function is still $f(x_1, \ldots, x_n) = R^2$.

It is now relatively easy to estimate both the gain and loss terms. For sufficiently large n, the offspring will have the distance $d = \sigma\sqrt{n-1}$ in the subsystem $x_2, \ldots, x_n$. On the x_1-axis, the offspring can lay on either side of the

[3]Constant factors, such as 1.5, 1.0, and 1/1.5, might work as well.

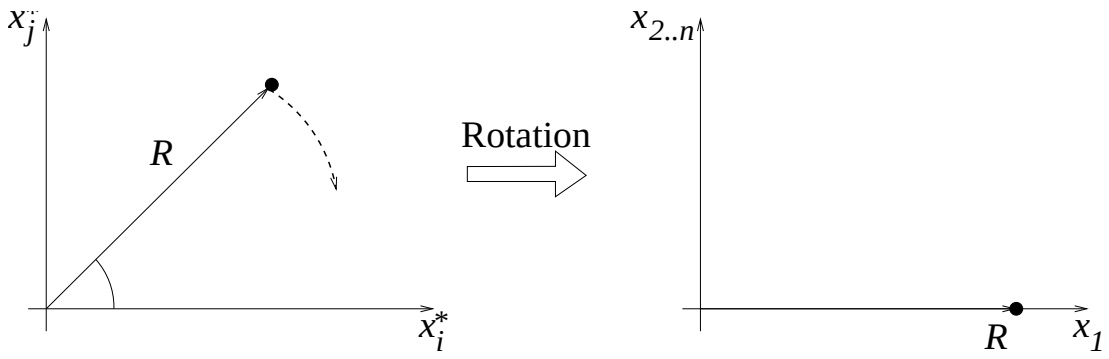

Fig. 1. The left figure shows the best parent with distance R to the optimum in some problem coordinates x_i^*. The right figure shows the situation after the application of a coordinate rotation such that in new coordinates, the parent is at $x_1 = R$ and $x_{2..n} = 0$.

parent ($x_1 = R$). The effective distance toward the origin (the optimum's location) depends on the selection scheme (i.e., comma or plus) and the number of both parents (μ) and offspring (λ), and are subsumed in the constant $c_{\mu\pm\lambda}$, thus having an effective step towards the origin of $\sigma c_{\mu\pm\lambda}$. Substituting these quantities into the rate-of-progress formula (2) leads to:

$$\begin{aligned} \varphi &= f(R) - f(R - \sigma c_{\mu\pm\lambda}, \sigma, \ldots, \sigma) \\ \varphi &= R^2 - \left((R - \sigma c_{\mu\pm\lambda})^2 + \sigma^2, \ldots, \sigma^2\right) \\ \varphi &= 2R\sigma c_{\mu\pm\lambda} - \sigma^2 c_{\mu\pm\lambda}^2 - \sigma^2(n-1) \\ \varphi &\approx 2\sigma R c_{\mu\pm\lambda} - n\sigma^2 \end{aligned} \tag{5}$$

as has been derived by Beyer. It should be mentioned again that the presented derivation is rather probabilistic than exact. But due to its simplifications, it can be applied easily to other than quadratic test cases.

The rate-of-progress formula (5) is a downwards-opened parabola. The optimal step size σ_{opt}=$Rc_{\mu\pm\lambda}/n$ depends on three parameters. For a chosen number of dimensions n=10 and a particular evolution strategy, $c_{1,10}$=1.54, Figure 2 presents the resulting normalized progress φ^*=φ/R^2 for varying distances R=1,10,100. The x-axis is in logarithmic scale.

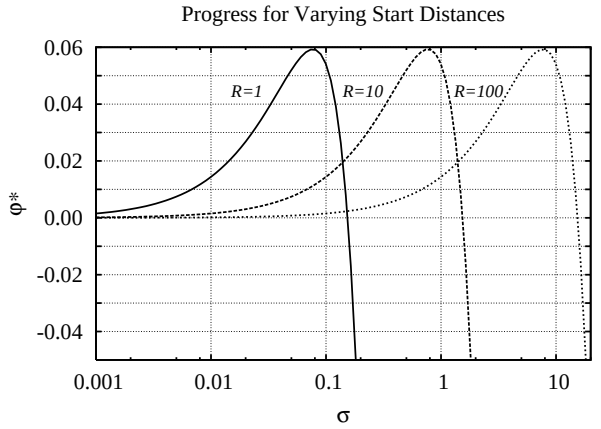

Fig. 2. This figure shows the normalized rate of progress a (1,10)-ES achieves on a quadratic objective function for varying distances R.

As can be seen, the procedure achieves reasonable progress only, if the step size σ is within an order of magnitude, for which Rechenberg has coined the term evolution window [12]. If considering σ^*=σ/R, all graphs would collapse into one.

III. Problem Description

Subsection II-B has shown that evolution strategies yield some suitable progress for step sizes σ within an order of magnitude. Only beyond a certain limit σ_l=$2\sigma_{\text{opt}}$, the progress is either zero ("plus"-strategy) or even negative ("comma"-strategy). But even in such cases, the dynamic adaptation mechanism would regulate the step size towards the optimal value $\sigma_{\text{opt}} = Rc_{\mu\pm\lambda}/n$ within a few subsequent generations resulting again in measurable progress.

However, the procedure might not be able to yield *positive* progress on all unimodal functions. As an example, Figure 3 presents some performance figures (averages over 10 independent runs) when applying various evolution strategies to the 10-dimensional function $f(x_1, \ldots, x_n) = \sum_i \sqrt{|x_i|}$. In all runs, the procedure started at $\vec{x}_0 = [10, 0, 0, \ldots, 0]^T$. It can be clearly seen that none of the evolution strategies yields any progress; the "plus" strategies stagnate, whereas the "comma" strategies degrade slightly.

Similar observations were made in the filed of evolutionary robotics [14]. There, various evolutionary algorithms frequently got stuck at suboptimal values, even though the fitness function was unimodal; it consisted only of quadratic terms.

It should be mentioned here that the problem could not be solved by increasing both the number of generations and/or the number of population members. However, when doing both increasing the population size very much *and* externally setting the step size σ to suitable values, some constant progress could be achieved. However, this last test was for debugging purposes only, since externally setting the step size needs to be dynamic during the course of optimization and requires knowledge about the optimum's location, which is actually the goal of the optimization procedure.

IV. Explanations

This section discusses the observed problem more from a theoretical point of view.

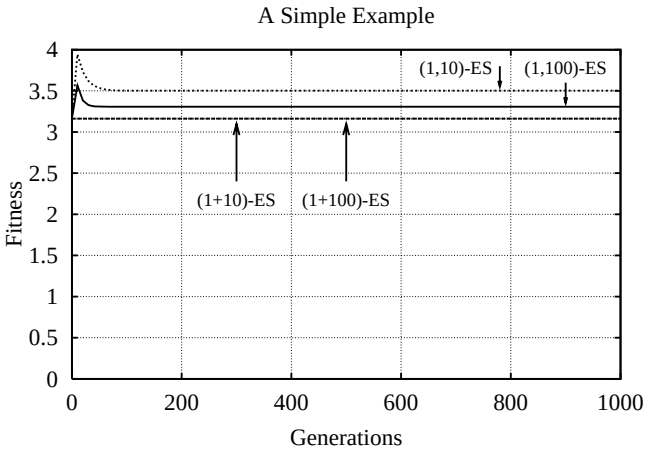

Fig. 3. This figure presents four different evolution strategies, (1,10)-ES, (1,100)-ES, (1+10)-ES, and (1+100)-ES trying to optimize the 10-dimensional function $f(x_1,\ldots,x_n)=\sum_i\sqrt{|x_i|}$. In all runs, the procedures started at $\vec{x}_0=[10,0,0,\ldots,0]^T$. It can be seen that "plus" strategies stagnate and that the "comma" strategies degrade slightly. Increasing the number of generations or increasing the number of offspring further would *not* bring *any* help.

The approach taken in Equation (5) considers a starting point $\vec{x}=[R,0,\ldots,0]^T$ and mutation vectors $\vec{Z}_g$ with $N(0,1)$-distributed components z_i. This approach is quite exact and has led to two terms, which have been approximated by gain=$2Rc_{\mu\pm\lambda}\sigma$ and loss=$n\sigma^2$.

The remainder of this paper considers objective functions of the form $f(\vec{x})=f(x_1,\ldots,x_n)=\sum_i f_i(x_i)$ with the constraint that all $f_{i=2\ldots n}()$ are equal. In this case, the two terms can be approximated by gain=$f_1(R)-f_1(R-c_{\mu\pm\lambda}\sigma)$ and loss=$nf_{i\neq1}(\sigma)-nf_{i\neq1}(0)$. If no simple expression in σ exists, the gain term can be approximated by gain$\approx f_1^{'}(x_1{=}R)c_{\mu\pm\lambda}\sigma$ quite well, as long as the objective function is smooth enough in the neighborhood of $x_1{=}R$. For example, for quadratic functions with $f_i{=}x_i^2$, this approximation would directly lead to gain=$2Rc_{\mu\pm\lambda}\sigma$, a term already seen in Equation (5).

This section is now turning to the objective function $f(x_1,\ldots,x_n){=}\sum_i\sqrt{|x_i|}$, which has already been investigated experimentally in Section III. Taking the approach discussed above, the rate-of-progress can be derived as follows (please note that all $f_i()$ are equal):

$$\begin{aligned}\varphi &\approx f_i^{'}(R)c_{\mu\pm\lambda}\sigma-nf_i(\sigma)\\ \varphi &\approx \frac{1}{2\sqrt{R}}c_{\mu\pm\lambda}\sigma-n\sqrt{\sigma}\ .\end{aligned}\qquad(6)$$

It can be easily seen that for all $0{\leq}\sigma{\leq}4n^2R/c_{\mu\pm\lambda}$, the approximated progress is negative. It should be noted that this interval is larger than the applicability of the approximation itself. The approach gain=$f_i^{'}(R)c_{\mu\pm\lambda}\sigma$ is valid only for small σ; for large σ, the gain should be calculated by the more exact formula gain=$f_1(R)-f_1(R-c_{\mu\pm\lambda}\sigma)$ leading to $\varphi\approx\sqrt{R}-\sqrt{R-c_{\mu\pm\lambda}\sigma}-n\sqrt{\sigma}$, which is *practically* negative for all $\sigma>0$. Figure 4 shows the normalized progress a (1,10)-ES achieves for varying distances R=1,10,100. Because of the negative progress, the evolution strategies try to minimize the loss, and hence, constantly decrease the step size σ, which in turn leads to a final fitness of $f\approx\sqrt{R}$ as was already seen in Figure 3.

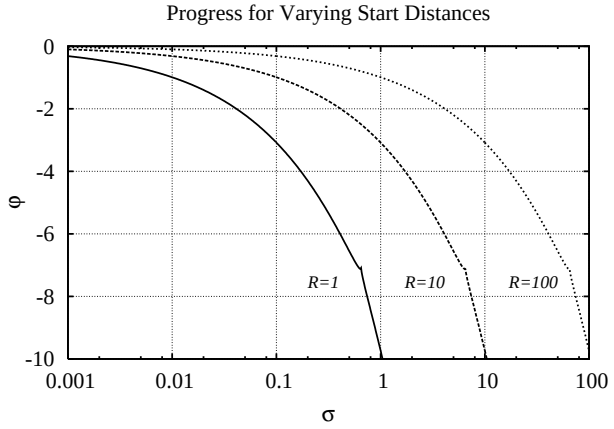

Fig. 4. This figure shows the normalized rate of progress a (1,10)-ES achieves on the function $f(x_1,\ldots,x_n)=\sum_i\sqrt{|x_i|}$ for varying distances R.

As indicated in the previous section, increasing the number of offspring tremendously and setting the step size to suitable values externally (thereby removing the step size self-adaptation mechanism) can yield measurable progress. Furthermore, it was indicated above, that the rate of progress is negative for just all *practical* $\sigma{>}0$. But a closer look at the rate-of-progress formula indicates that it has another maximum with the potential of a *positive* progress. To this end, Figures 5 and 6 show the progress some chosen evolution strategies achieve on the two and three-dimensional version of the objective function, respectively. It can be seen that on the two-dimensional version, even a (1,10)-ES yields some progress. However, when optimizing the three-dimensional instance, already a (1,10000)-ES is required. Increasing the number of dimensions further, requires a drastic increase in the number of

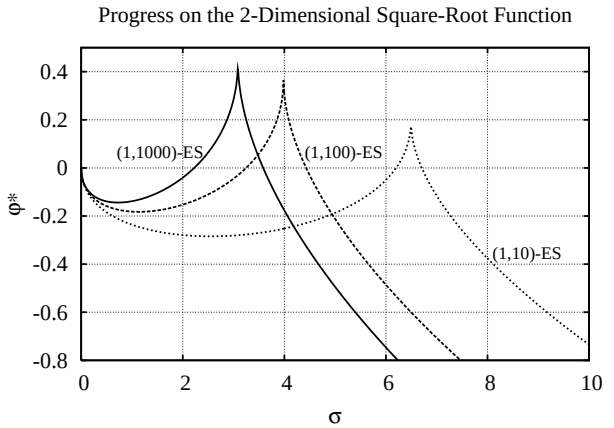

Fig. 5. This figure shows the normalized rate of progress various evolution strategies achieve on the two-dimensional objective function $f(x_1,x_2)=\sum_i\sqrt{|x_i|}$.

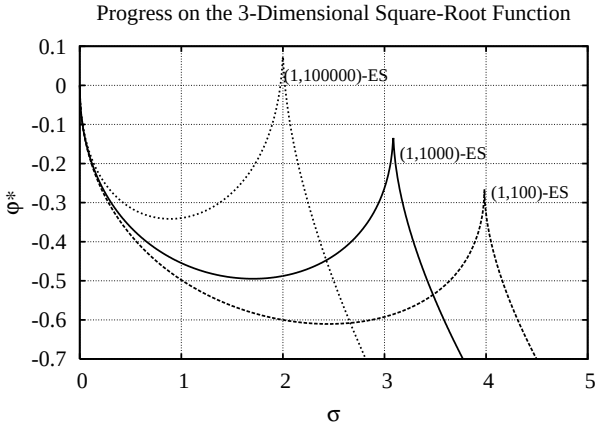

Fig. 6. This figure shows the normalized rate of progress various evolution strategies achieve on the three-dimensional objective function $f(x_1, x_2, x_3) = \sum_i \sqrt{|x_i|}$. The achievable progress of the (1,10000)-ES is estimated, since the exact value for $c_{1,10000}$ is not provided by the literature.

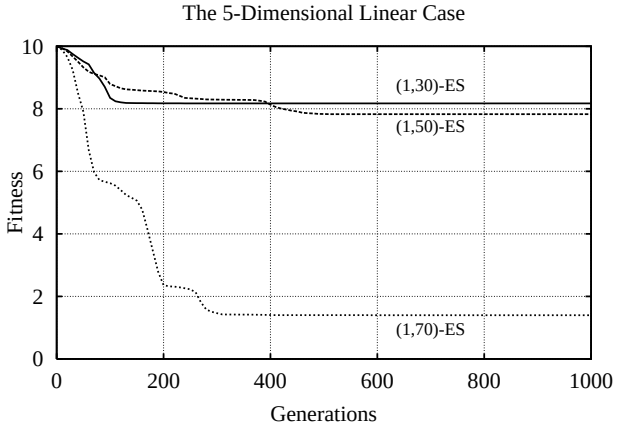

Fig. 7. This figure presents the performance (averages over 10 independent runs) of three different evolution strategies, (1,30)-ES, (1,50)-ES, (1,70)-ES, optimizing the 5-dimensional objective function $f(x_1, \ldots, x_5) = \sum_i |x_i|$. In all runs, the procedures started at $\vec{x}_0 = [10, 0, 0, \ldots, 0]^T$.

offspring. Even though some progress is achievable in theory, practical considerations exclude any application for even a moderate number of dimensions; that is the *curse* of high-dimensional optimization, because the required population sizes are growing too fast.

Figures 5 and 6 allow for another observation. Even a (1,100)-ES, for example, may fail at the two-dimensional case (Figure 5, middle graph). If the initial step size is below σ=1, the self adaptation mechanism would regulate the step size towards maximal performance. Unfortunately, in this case, this would be at σ=0-end of the graph, i.e., the left-hand-side; the self-adaptation process would not regulate the step size towards the other optimum at $\sigma = 4$. In other words, depending on the initial value, the procedure would constantly reduce the step size resulting in zero progress, which would be furthermore independent of the actual distance R. The procedure would then prematurely converge for obvious reasons.

V. Extension

This section extends some of the results to other fitness functions as well as algorithms.

A. Polynomials $|x|^p$ with p=1

The derivation of the rate of progress indicates that objective functions of the form $f(\vec{x})=\sum_i |x_i|^p$ yield the terms gain$\approx pR^{p-1}c_{\mu\pm\lambda}\sigma$ and loss=$n\sigma^p$ arriving at the following rate of progress:

$$\varphi \approx pR^{p-1}c_{\mu\pm\lambda}\sigma - n\sigma^p \ . \qquad (7)$$

Equation (7) indicates that p=1 is the critical value at which the curse of high-dimensional optimization occurs. For p>1, the rate of progress is always positive for appropriate values of the step size σ. For the case p=1, the rate of progress approximates to $\varphi=c_{\mu\pm\lambda}\sigma - n\sigma=(c_{\mu\pm\lambda} - n)\sigma$, which is positive as long as $c_{\mu\pm\lambda}>n$ holds. While it is rather easy to optimize such functions for small dimensions n, the curse is again that the number of required offspring grows too fast with the number of dimensions. The published $c_{\mu\pm\lambda}$-values [12] suggest that increasing the number of offspring by roughly an *order of magnitude*, merely increases $c_{\mu\pm\lambda}$ by the value 1. A rough estimate suggests that the number of offspring $\lambda\approx 10^n$ grows exponentially in the number of dimensions n of the search space. This means that also the number $O(\exp n)$ of function evaluations is exponential in n.

Figures 7 and 8 show some performance figures of various evolution strategies on a 5-dimensional and a 7-dimensional version of $f(\vec{x})=\sum_i |x_i|$, respectively. It can be seen that the required number of offspring increases significantly (resulting in a very poor performance). It should be mentioned here that the probabilistic description of the rate of progress (Equation (7)) is not vary accurate, since the number of dimensions n is relatively small (significantly below 20 or so). It should be

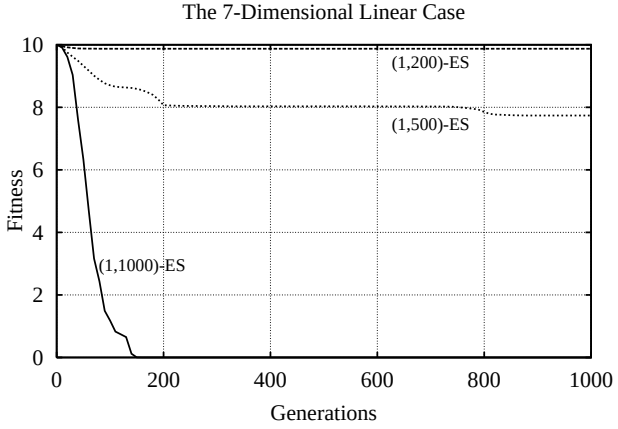

Fig. 8. This figure presents the performance (averages over 10 independent runs) of three different evolution strategies, (1,200)-ES, (1,500)-ES, (1,1000)-ES, optimizing the 7-dimensional objective function $f(x_1, \ldots, x_7) = \sum_i |x_i|$. In all runs, the procedures started at $\vec{x}_0 = [10, 0, 0, \ldots, 0]^T$.

noted that the 10-dimensional case is another good example of the curse of high-dimensional optimization, because a (1,10000)-ES would not be sufficient.

B. Real-World Objective Functions

It might be argued that the presented objective functions are somewhat atypical, since almost all (practically relevant real-world) objective functions have a quadratic shape around the optimum $\vec{x}_o$. In such a case, the objective function would change to $f_1(x_1)=\sqrt{|x_1|}$ for $x_1 \approx R$ and $f_{i=2..n}(x_i)=x_i^2$. For a starting point $\vec{x}_0=[R,\ldots,0]$, the gain and loss terms would arrive at gain$\approx c_{\mu\pm\lambda}\sigma/(2\sqrt{R})$ and loss$=n\sigma^2$. With an optimal step size $\sigma_{\text{opt}}=c_{\mu\pm\lambda}/(4n\sqrt{R})$ the achievable progress is $\varphi=c_{\mu\pm\lambda}^2/(16Rn)$, which is inversely proportional to the distance R. For such functions, the progress is always positive for optimal step sizes. However, the progress assumes very small values for large search spaces and large distances R.

Noise or noisy fitness evaluation is another factor to be considered in real-world applications. In such cases, the loss term increases to loss$=n\sigma^2+n\epsilon^2$, with ϵ denoting the *effective* noise level. It is obvious that the loss exceeds the gain term for sufficiently large distances R (please, remember that the gain is inversely proportional to $\sqrt{R}$). Hence, the experiments with such objective functions, which are given by the real-world system, might fail and give the impression that the procedure might be getting stuck at distracting local optima. But it would be the curse of the too small a gain within the high-dimensional search space. Such observations have been reported in the evolution of an artificial insect eye [14].

C. Genetic Algorithms

This paper has clearly focused on evolution strategies. One might argue that the presented problems and explanations do not apply to genetic algorithms. This is both true and false to some extent. Genetic algorithms typically apply mutations with a small probability $p_m = 1/n$ and apply various recombination operators with a rather high probability $p_r = 0.95$ ([3]). The optimization of the $\sqrt{x_i}$-function can be interpreted as going along a narrow valley. If this valley is not along one of the coordinate axes but is rather pointing into the n-dimensional search space, the probability of such a mutation is approaching $p = 1/(n^n)$, which is less than exponentially small. Only if the valley is oriented along one (or a few) axes, the resulting performance is of practical relevance. Furthermore and contrary to widely held beliefs, recombination helps only in rare cases [8], [13].

VI. Conclusions

This paper has presented some background material on the convergence of evolution strategies. Starting off with some unexpected behavior, i.e., premature convergence at suboptimal values in simple, continuous, unimodal functions, this paper has extended existing theoretical approaches to such functions.

The sound application of these theoretical methods is limited, since the entire function is not always rotationally invariant (even though, and this is quite important, evolution strategies with one individual-specific global step size behave so). Nevertheless, these analyses give some good indications about the observed behavior and their underlying reasons. Some presented performance tests have supported the derived descriptions of the achievable progress.

In summary, evolution strategies might get stuck in unimodal objective functions. For some of them, the high-dimensional search space is one of the major reasons; here, the loss simply exceeds the attainable gain.

References

[1] T. Bäck, U. Hammel, and H.-P. Schwefel, Evolutionary Computation: Comments on the History and Current State. *IEEE Transactions on Evolutionary Computation*, 1(1), 1997, 3-17.

[2] H.-G. Beyer. *The Theory of Evolution Strategies.* Springer-Verlag Berlin, 2001.

[3] K.A. De Jong. An Analysis of the Behavior of a Class of Genetic Adaptive Systems. Ph.D. Thesis, University of Michigan, 1975.

[4] S. Droste, Th. Jansen, I. Wegener. On the Optimization of Unimodal Functions with the (1+1) Evolutionary Algorithm. *Proceedings of The Fifth International Conference on Parallel Problem Solving from Nature (PPSN V).* Springer-Verlag, Berlin, 13-22, 1998.

[5] D.B. Fogel. *Evolutionary Computation: Toward a New Philosophy of Machine Learning Intelligence.*. IEEE Press, NJ, 1995.

[6] L.J. Fogel. "Autonomous Automata", *Industrial Research.* **4**:14-19, 1962.

[7] D.E. Goldberg. *Genetic Algorithms in Search, Optimization and Machine Learning.* Addison-Wesley, Reading, MA, 1989.

[8] T. Jansen and I. Wegener. On the Analysis of Evolutionary Algorithms – A Proof That Crossover Really Can Help. In: J. Nešetřil, ed., *Proceedings of the 7th Annual European Symposium on Algorithms (ESA '99*, Springer-Verlag, Berlin, Germany, 184-193, 1999.

[9] D.G. Luenberger. *Linear and Nonlinear Programming.* Addison-Wesley, Menlo Park, CA, 1984.

[10] H. Mühlenbein and D. Schlierkamp-Voosen. The Science of Breeding and its Application to the Breeder Genetic Algorithm. *Evolutionary Computation.* **1**(4):335-360, 1993.

[11] W.H. Press, B.P. Flannery, S.A. Teukolsky, and W.T. Vetterling. *Numerical Recipes.* Cambridge University Press, 1987.

[12] I. Rechenberg, *Evolutionsstrategie* (Frommann-Holzboog, Stuttgart, 1994).

[13] R. Salomon. Reevaluating Genetic Algorithm Performance under Coordinate Rotation of Benchmark Functions; A survey of some theoretical and practical aspects of genetic algorithms. *BioSystems.* **39**(3):263-278, 1996.

[14] R. Salomon and L. Lichtensteiger, Exploring different Coding Schemes for the Evolution of an Artifi cial Insect Eye. *Proceedings of The First IEEE Symposium on Combinations of Evolutionary Computation and Neural Networks*, 2000, 10-16.

[15] H.-P. Schwefel. *Evolution and Optimum Seeking.* John Wiley and Sons, NY. 1995.

[16] D. Wolpert and W. Macready. No Free Lunch Theorems for Optimization. *IEEE Transactions on Evolutionary Computation.* **1**(1):67-82, 1997.