EVOLUTION STRATEGIES

Guenter Rudolph, chair

On The Convergence Properties of a Simple Self-Adaptive Evolutionary Algorithm

John DeLaurentis
Sandia National Laboratories
P. O. Box 5800, MS 1110
Albuquerque, NM 87185-1110
jdlauren@sandia.gov

Lauren Ferguson
Texas Technical University
alaferg@yahoo.com

William E. Hart
Sandia National Laboratories
P. O. Box 5800, MS 1110
Albuquerque, NM 87185-1110
wehart@sandia.gov

Abstract

We consider the convergence properties of self-adaptive evolutionary algorithms (EAs). The self-adaptive search component of these EAs *implicitly* adapts the step lengths in response to their efficacy for generating improving points. We analyze the convergence of a $(1, \lambda)$-EA with simpler mutation updates than are commonly used in Evolutionary Strategies or Evolutionary Programming methods. Although self-adaptive EAs have been analyzed by several authors, our analysis provides the first exact proof of convergence for an implicitly self-adaptive EA. Our experimental and theoretical analysis demonstrates that this EA robustly converges to the optimum of a symmetric, unimodal problem.

1 INTRODUCTION

The distinguishing feature of self-adaptive evolutionary algorithms (EAs) is that the control parameters are evolved by the evolutionary algorithm. This is particularly important when using EAs to optimize over continuous design spaces, since an effective search requires a search over different neighborhoods of the domain as well as refined search at different length-scales within interesting neighborhoods [6]. Thus self-adaptation is a central feature of EAs like evolutionary stategies (ES) and evolutionary programming (EP), which are applied to continuous design spaces. Eiben et al. [6] distinguishes between *explicit self-adaptation*, in which the success of previous iterations is explicitly employed to adapt the step length, and *implicit self-adaptation*, in which the the step lengths are evolved along with the search parameters. Eiben et al. [6] distinguishes these forms of self-adaptation by denoting explicit methods as *adaptive* EAs and denoting implicit methods as *self-adaptive* EAs. However, this distinction between adaptive and self-adaptive methods does not appear to have been widely adopted.

Explicitly self-adaptive EAs have been analyzed by a number of authors, and a variety of analyses have proven convergence theories for different explicitly self-adaptive formulations [1, 7, 10, 9, 11, 12, 13, 15]. By contrast, Beyer [3, 4] has developed the only theoretical investigation of implicitly self-adaptive EAs; he considers the convergence of the implicitly self-adaptive (μ, λ)-ES. Beyer notes that EAs can be described by an inhomogeneous Markovian process, and that the stochastic evolution of the system can be expressed by Chapman-Kolmogorov equations. However, he further notes that a direct treatement of these equations is generally quite difficult, and thus his analysis treats the (μ, λ)-ES as a dynamical system from which simpler dynamical systems are derived and validated.

In this paper we reconsider the convergence properties of the implicitly self-adaptive $(1, \lambda)$-ES. We simplify this EA's dynamics by considering a mutation operator that employs a discrete random variable. In this EA, there are a finite (and small) number of possible individuals that can be generated in each iteration. Consequently, the expected behavior of the EA can be characterized from one iteration to the next. Our analysis provides a convergence theory for the $(1, \lambda)$-ES for one-dimensional symmetric, unimodal objective functions. Although this is clearly an artificial class of objective functions, we hope that the techniques used in this analysis will be applicable to broader problem domains.

A complete description of our analysis is beyond the scope of this paper, so we refer the reader to DeLaurentis et al. [5] for further details. In addition to our convergence analysis, we have also identified param-

eters for which this convergence theory is practically relevant, and we provide some simple comparisons of these EAs with an $(1, \lambda)$-ES using the standard log-normal mutation operator. Finally, we describe how this convergence theory can be exploited to show how non-elitist self-adaptive $(1, \lambda)$-EAs can fail to robustly converge to globally optimal solutions. This result follows from the convergence theory that we have proven, and thus we believe that this is a broader property of implicitly self-adaptive EAs.

2 BACKGROUND

Perhaps the most common approach to the analysis of ESs is to the consider the *progress rate*, φ. Typical progress rates reflect the expected change of the ES from one iteration to the next with respect to a progress metric, which reflects the distance of a point from a given local optimum. Beyer [3] considers the progress rate of a standard self-adaptive $(1, \lambda)$-ES with log-normal mutation and concludes:

> Furthermore, applying the scaling rule $\tau = c_{1,\lambda}\sqrt{N}$ *ensures the linear convergence* of the ES algorithm. (His italics)

Beyer models the $(1, \lambda)$-ES with an approximate noisy map, and his analysis considers both first-order and second-order dynamics of this ES.

Although this analysis provides significant insight into the dynamics of this ES, it does not provide an exact convergence theory of this ES. Beyer's analysis makes a variety of nontrivial approximations to simplify the stochastic process underlying this ES. For example, the approximations that Beyer makes partially decouples the interaction between the step-length parameter and position parameter, which may fundamentally affect the convergence of these random variables; variations in the step-length parameters are modeled with normally distributed noise, which does not depend upon the position parameters. Further, Beyer's analysis does not carefully characterize the type of stochastic convergence exhibited by the approximate noisy map. Although the analysis of first- and second-order dynamics suggests that this ES has linear convergence, Beyer's results do not state this result in terms of standard forms of stochastic convergence (e.g. almost sure, in probability, etc.) [8].

Thus it is clear that an exact a convergence theory has not been previously developed for any class of implicitly self-adaptive EAs. The convergence theory that we describe considers the sequence of best points found in each iteration of an implicitly self-adaptive $(1, \lambda)$-ES, and we show that these points converge *almost surely* (i.e. with probability one). If Y and Y_t are random variables, then we say that the sequence $\{Y_t\}_{t\geq 0}$ converges almost surely to Y if $P\{\lim_{t\to\infty} Y_t = Y\} = 1$. We write this as $Y_t \xrightarrow{a.s.} Y$. See Grimmett and Stirzaker [8] for a thorough discussion of stochastic convergence.

3 ANALYSIS OF A SIMPLE ES

3.1 OVERVIEW

In this section we describe a class of self-adaptive EAs that are guaranteed to converge on one-dimensional, unimodal objective functions. We consider a simplified $(1, \lambda)$-ES that generates λ new points in each iteration and selects the best point generated for the next iteration. Figure 1 describes Algorithm A, which updates the mutation scale with the standard update rule: $\sigma_t^i = \sigma_{t-1} \cdot D_t^i$. However, this EA is distinguished by the fact that it uses a discrete random variable for D_t^i, as well as the discrete random variable, B_t^i, to generate x_t^i.

Given x_0, σ_0
For $t = 1, \ldots$
  For $i = 1 : \lambda$
    $\sigma_t^i = \sigma_{t-1} \cdot D_t^i$
    $x_t^i = x_{t-1} + \sigma_t^i \cdot B_t^i$
  End
  $j = \arg\min_{i=1:\lambda} \ f(x_t^i)$
  $x_t = x_t^j$
  $\sigma_t = \sigma_t^j$
End

Figure 1: Algorithm A: A self-adaptive $(1, \lambda)$-ES for one-dimensional problems. The random variables D_t^i and B_t^i are discrete random variables described in the text.

Let d_t^i be the realization of the random variable D_t^i: $D_t^i \in \{\gamma, 1, 1/\gamma\}$, $1/2 < \gamma < 1$. Let $\nu_1 = P\{D_t^i = \gamma\}$, $\nu_2 = P\{D_t^i = 1\}$ and $\nu_3 = P\{D_t^i = 1/\gamma\}$ for all t; we assume that these probabilities are nonzero. Thus $\sigma_t^i = \sigma_{t-1} d_t^i$. A step length σ_t^i is used to generate the point $x_t^i = x_{t-1} + \sigma_t^i \cdot b_t^i$, where b_t^i is the realization of the random variable B_t^i: $B_t^i \in \{-1, +1\}$ with probabilities $\{\frac{1}{2}, \frac{1}{2}\}$ respectively. Each point x_t with its corresponding step length σ_t becomes the parent point for the next iteration.

Let X_t^λ and Σ_t^λ be random variables that describe the distribution of the values of x_t and σ_t respectively when a population of size λ is used by Algorithm A. Let $\mathcal{F}_t$ be the sequence of σ-algebras that describe the random events that underly X_t^λ and Σ_t^λ. The key observation that underlies our analysis is that the convergence of X_t^λ and Σ_t^λ are complementary. For example, Σ_t^λ tends to grow when X_t^λ is far from the optimum, but in this situation X_t^λ is moving toward the optimum as much as Σ_t^λ grows. Similarly, when X_t^λ is very close to the global optimum then X_t^λ may need to move away from the optimum. But in this case, the value of Σ_t^λ is likely to decrease.

This observation led us to consider the convergence of the random variable $Z_t^\lambda = |X_t^\lambda| + W_\gamma \Sigma_t^\lambda$, where

$$W_\gamma = \frac{3\gamma + 1}{4(1-\gamma)}.$$

Our analysis applies when Algorithm A is applied to objective functions that satisfy the following assumption:

Assumption 1 *The function $f : \mathbf{R} \to \mathbf{R}$ has the property that*

1. *There exists a unique global minimum $x^* = 0$,*
2. *f is symmetric about x^* (i.e. $f(x) = f(2x^* - x)$), and*
3. *f is monotonically increasing for $x \in (x^*, \infty)$.*

Note that we assume that $x^* = 0$ only for convenience sake, since if an EA converges on a function that satisfies this condition, then we can show convergence for any other function h with nonzero global optimizer by optimizing the function $f(x) = h(x + x^*)$. Assumption 1 requires that f be unimodal, but it is quite weak otherwise. Assumption 1 does *not* require that f be continuous, and the global optimum can be at an isolated point (e.g., see Figure 2). Finally, note that since f is monotonically increasing for $x \in (x^*, \infty)$ then because of symmetry f is monotonically decreasing for $x \in (-\infty, x^*)$.

3.2 ANALYSIS

Our analysis begins by showing that Z_t^λ converges almost surely for sufficiently large λ. Given this, we show that the step lengths Σ_t^λ converge to zero, and finally we show that this implies that X_t^λ converges to x^*. A random process X_t is a super-martingale if $E[|X_t|] < \infty$ and $E[X_{t+1}|\mathcal{F}_t'] \leq X_t$, where $\mathcal{F}_t'$ is the family of σ-algebras that describe the events underlying X_t [8]. The following lemma provides the key result for our proof of Proposition 1.

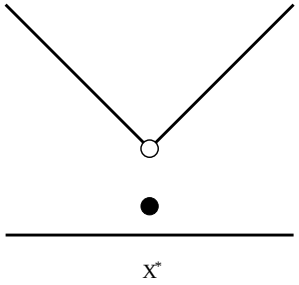


Figure 2: An example of a function satisfying Assumption 1 with an isolated global minimum.

Lemma 1 *Let $\frac{1}{2} < \gamma < 1$, and suppose that f satisfies Assumption 1. There exists $\lambda_0 > 0$ such that for all $\lambda \geq \lambda_0$, $E(Z_{t+1}^\lambda \mid \mathcal{F}_t) \leq Z_t^\lambda$.*

The following proposition shows that Z_t^λ is a super-martingale, which roughly shows that Z_t^λ decreases on average.

Proposition 1 *Let $\frac{1}{2} < \gamma < 1$, and suppose that the function f satisfies Assumption 1. Then there exists $\lambda_0 > 0$ such that for all $\lambda \geq \lambda_0$, Z_t^λ is a super-martingale with respect to the σ-algebras $\mathcal{F}_t$.*

Proof. Note that $E(Z_t^\lambda) < \infty$, since there are a finite number of states that can be reached by Algorithm A after t iterations, and Z_t^λ is finite for each of these states. From Lemma 1 we know that $E(Z_{t+1}^\lambda \mid \mathcal{F}_t) \leq Z_t^\lambda$. Together, these results show that Z_t^λ is a super-martingale with respect to $\mathcal{F}$. ∎

In the following results, we use the value λ_0 described by Proposition 1. The following corollary follows immediately from the fact that Z_t^λ is a nonnegative super-martingale.

Corollary 1 *Let $\frac{1}{2} < \gamma < 1$, and suppose that the function f satisfies Assumption 1. For all $\lambda \geq \lambda_0$, there exists a random variable Z_∞^λ such that $Z_t^\lambda \xrightarrow{a.s.} Z_\infty^\lambda$.*

Corollary 1 ensures that X_t^λ and Σ_t^λ almost surely generate a convergent sequence. This result confirms the observation noted above: X_t^λ and Σ_t^λ converge in a complementary fashion. However, this result is not sufficient to demonstrate that both of these random variables converge to zero. The following theorem uses Corollary 1 to show that Σ_t^λ converges to zero.

Theorem 1 *Let $\frac{1}{2} < \gamma < 1$, and suppose that the function f satisfies Assumption 1. For all $\lambda \geq \lambda_0$, $\Sigma_t^\lambda \xrightarrow{a.s.} 0$.*

Finally, we prove our main result: X_t^λ converges to zero almost surely. The following technical assumption is required for this analysis.

Assumption 2 *ν_1, ν_3 and λ are chosen so that*

$$1 - \left(1 - \frac{\nu_3}{2}\right)^\lambda + \left(\frac{\nu_3}{2}\right)^\lambda \geq \left(\frac{1+\nu_1}{2}\right)^\lambda - \left(\frac{1-\nu_1}{2}\right)^\lambda.$$

Theorem 2 *Let $\frac{1}{2} < \gamma < 1$, and suppose that the function f satisfies Assumption 1. Further, let ν_1, ν_3 and λ satisfy Assumption 2. There exists $\lambda_1 \geq \lambda_0$ such that for all $\lambda \geq \lambda_1$, $X_t^\lambda \xrightarrow{a.s.} 0$.*

4 PRACTICAL RELEVANCE

In this section, we consider the practical relevance of this convergence theory in cases where $\lambda \leq 5$. Note that Theorem 2 is not practically relevant when $\lambda \geq 6$. In this case, the stochastic sampling performed by Algorithm A is unnecessessary, since it is at least as efficient to simply enumerate the six possible new points that can be generated in each iteration. The following section considers the range of parameters for Algorithm A for which the convergence theory applies, particular when $\lambda = 5$. Subsequently, we consider some simple experiments that confirm that the performance of Algorithm A is roughly comparable to a standard self-adaptive ES using these parameters.

4.1 FEASIBLE PARAMETERS

We consider values of ν_i and γ for which the convergence theory for Algorithm A (in Proposition 1 and Theorems 1 and 2) applies for values of $\lambda < 6$. The following lemma makes it clear that the convergence theory does *not* apply for all possible values of ν_i.

Lemma 2 *Suppose that $\lambda \leq 5$. Then there exists $\epsilon > 0$ such that if $\nu_1 < \epsilon$ then Z_t^λ is not a super-martingale.*

Proof. To ensure that Z_t^λ is a super-martingale, we must have

$$E(Z_{t+1}^\lambda \mid \mathcal{F}_t) \leq Z_t^\lambda \tag{1}$$

for all possible values of Z_t^λ. Consider the case where Algorithm A is within $\sigma_{t-1}/(2\gamma)$ of the optimum. In this case, the six possible values of $|X_{t+1}^\lambda|$ that can be realized are worse than $|X_t^\lambda|$. Further, of the six terms in the expectation, only the terms representing contraction steps can reduce the value of Z_{t+1}^λ. Thus if Equation (1) is satisfied, the probability of the contraction steps, ν_1, cannot be arbitrarily small. But this is what we have assumed, so X_t^λ is not a super-martingale for sufficiently small values of ϵ. ∎

It is not clear that a simple analytic characterization can be developed to describe the different feasible choices for ν_i and γ. Consequently, we have numerically evaluated the set of choices for ν_i and γ for which our convergence theory is applicable. Our numerical model considers the ν_i and γ that satisfy (a) Assumption 2, (b) the constraint $\nu_1 + \nu_3 < 1$, and (c) the four main cases of the analysis in Proposition 1:

1. $|x_t + \sigma_t \gamma b| < |x_t|$,
2. $|x_t + \sigma_t b| < |x_t|$,
3. $|x_t + \sigma_t b/\gamma| < |x_t|$, and
4. $|x_t + \sigma_t \gamma b| \geq |x_t|$.

where $x_t + \sigma_t db$ is the best value of x_{t+1} possible for some $d \in \{\gamma, 1, 1/\gamma\}$ and $b \in \{-1, 1\}$. Note it is a simple matter to show that if case (4) does not hold then one of the other three cases holds. For each of these four cases we consider the expectation in Equation 1, which imposes a constraint on the values of ν_i and γ. In fact, this equation imposes several constraints on ν_i and γ, one for each possible rank-ordering of the six possible values of x_{t+1}. The details of how these contraints are derived is discussed in a technical report [5].

We enumerated feasible values of ν_i and γ with respect to these constraints. For these calculations, we assumed that no ties could occur in the rank-order of possible values of x_{t+1} (for example, this is always true if x_0 is irrational). Further, we fixed γ and sampled ν_1 and ν_3 on a fine mesh. The results of these calculations are shown in Figure 3. For values of $\gamma > 1/\sqrt{2}$ we did not find any feasible parameters in our calculations, which may simply reflect the weakness of our approximations in this case. However, if we assume that $\gamma < 1/\sqrt{2}$ then our results clearly indicate that there is a wide range of feasible parameters for Algorithm A. Further, there is significant overlap between these figures, which suggests that there may be values of ν_1 and ν_3 for which the convergence theory is valid for $0.5 < \gamma < 1/\sqrt{2}$.

4.2 EXPERIMENTAL COMPARISONS

We compared Algorithm A with a standard self-adaptive $(1, \lambda)$-ES on several simple unimodal problems. These experiments provide further confirmation

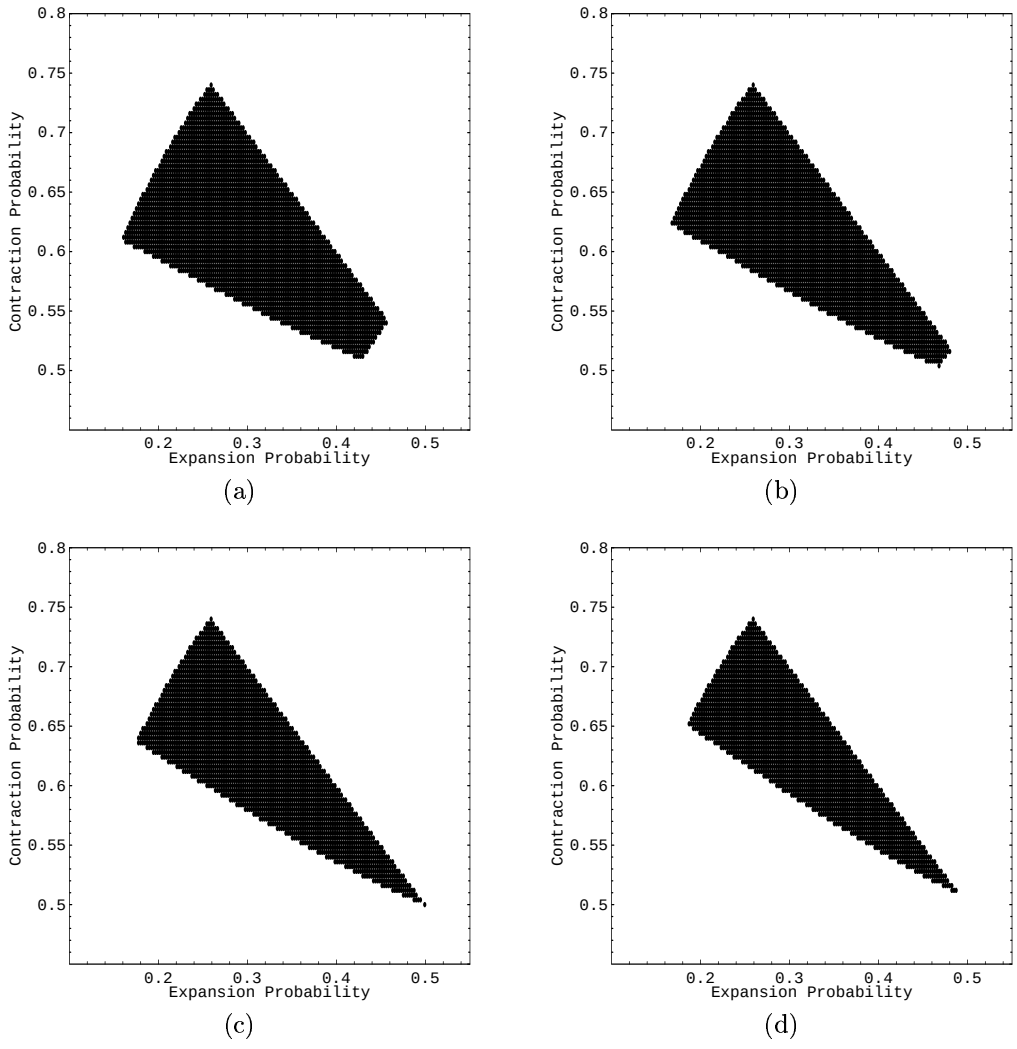


Figure 3: Domains of feasibility for ν_1 and ν_3 for (a) $\gamma = 0.55$, (b) $\gamma = 0.60$, (c) $\gamma = 0.65$ and (d) $\gamma = 0.70$. The axes are the expansion probability, ν_3, and the contraction probability, ν_1.

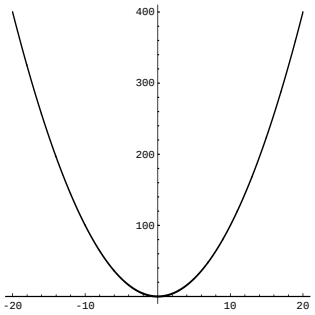
of the convergence properties of Algorithm A. Further, they demonstrate that Algorithm A can have comperable performance to standard self-adaptive EAs, despite the fact that Algorithm A uses a discrete random variables in the step-length adaptation and in the generation of mutation steps.

The following test functions were used in our experiments:

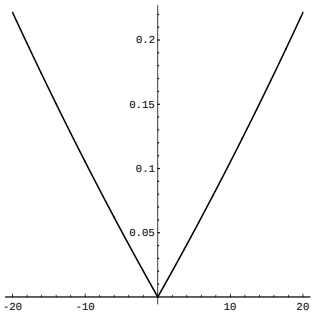
- $f_1(x) = x^2$
- $f_2(x) = e^{|x|/100} - 1$
- $f_3(x) = \sqrt{|x|}$
- $f_4(x) = 10 \lceil |x| \rceil + |x|$

All of these functions satisfy Assumption 1. The function f_1 is a standard test problem. Function f_2 becomes much steeper than f_1, which stresses some aspects of the convergence theory. Function f_3 is nonconvex, and function f_4 is discontinuous in regular intervals (and at the global optimum). These functions are illustrated in Figure 4.

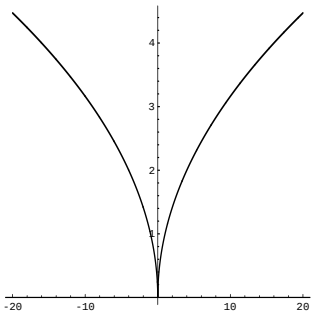
We compared Algorithm A with Algorithm B, a standard self-adaptive ES using log-normal self-adaptation of the step lengths and normally distributed mutation steps (e.g. see [2, 4]). Figure 5 describes Algorithm B; $N(0,1)$ refers to a normally distributed random variable with mean zero and variance one. Note that this algorithm is identical to Algorithm A, except for the updates to σ_t^i and x_t^i, which simply employ different random variables in place of D_t^i and B_t^i.



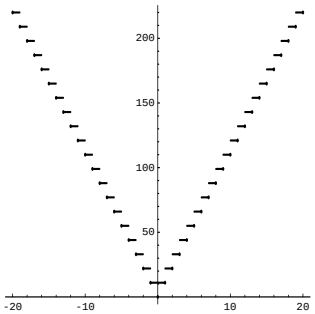
(a)



(b)



(c)



(d)

Figure 4: Graphs of functions (a) f_1, (b) f_2, (c) f_3 and (d) f_4.

Given x_0, σ_0
For $t = 1, \ldots$
  For $i = 1 : \lambda$
    $\sigma_t^i = \sigma_{t-1} \cdot \exp(N(0,1))$
    $x_t^i = x_{t-1} + \sigma_t^i \cdot N(0,1)$
  End
  $j = \arg\min_{i=1:\lambda} \ f(x_t^i)$
  $x_t = x_t^j$
  $\sigma_t = \sigma_t^j$
End

Figure 5: Algorithm B: A standard self-adaptive $(1, \lambda)$-ES for one-dimensional problems.

Both algorithms were run with 1000 different random seeds on each test function. Each optimizer has a single minimum at $x = 0$ with $f(0) = 0$. The optimizers were terminated after the function evaluation was less than 10^{-8}, except for f_4; f_4 has an isolated minimum at $x = 0$ and the function approaches 10 as x goes to zero. Thus we terminated these experiments when the optimizers found points whose function evaluation was less than $10 + 10^{-8}$. Further, we consider the behavior of these algorithms for two different initial conditions: (a) $x_0 = 10$ and $\sigma_0 = 100$ and (b) $x_0 = 1000$ and $\sigma_0 = 100$. Preliminary experiments suggested that Algorithm B worked well on these problems by effectively contracting the step length. Consequently, Algorithm A was run with $\nu_3 = 0.25$, $\nu_1 = 0.7$ and $\gamma = 0.55$.

Tables 1 and 2 compare the mean number of function evaluations before termination for these two initial conditions. Additionally, these tables show the results of a pairwise-comparison between these algorithms: for each random seed we compare the number of function evaluations needed for each algorithm and we report the percentage of trials for which Algorithm A is better.

Function	Mean Evals A	Mean Evals B	Pairwise Score
f_1	305.8	281.7	53.9
f_2	376.7	361.4	57.1
f_3	795.7	736.4	67.3
f_4	458.5	447.0	63.1

Table 1: Comparison of Algorithms A and B on the test functions when $x_0 = 1000$. The pairwise score is the percentage of trials that Algorithm A terminated after fewer function evaluations than Algorithm B.

Function	Mean Evals A	Mean Evals B	Pairwise Score
f_1	174.2	224.1	71.5
f_2	242.4	306.7	73.7
f_3	649.5	690.8	84.3
f_4	312.1	394.6	78.8

Table 2: Comparison of Algorithms A and B on the test functions when $x_0 = 10$. The pairwise score is the percentage of trials that Algorithm A terminated after fewer function evaluations than Algorithm B.

These results confirm the robustness of Algorithm A; this method found near-optimal points in every random trial of the experiments. Further, these experiments suggest that Algorithm A can perform as efficient a search as a standard self-adaptive EA. In fact, the experiments indicate that Algorithm A may me slightly more efficient, but these results are too preliminary to draw conclusions. Finally, we note that the total run-time of Algorithms A and B appears to be correlated with the slope of the function near the origin; when the slope is higher the run-time is longer. We conjecture that this reflects the fact that the step length needs to be contracted to a smaller scale to ensure convergence on narrower local minima.

5 GLOBAL CONVERGENCE

The analysis in Section 3 provides a concrete basis for expecting robust behavior from a self-adaptive $(1, \lambda)$-ES. This analysis also provides insight into the question of whether self-adaptive EAs are guaranteed (with probability one) to converge to globally optimal solutions in all cases. Rudolph [14] illustrates how self-adaptive, elitist EAs can fail to converge to globally optimal solutions with probability one. However, the EA that Rudolph considers uses an explicit self-adaptive control mechanism: whenever there is an improving mutation the step length is increased and it is decreased otherwise. By contrast, Algorithm A uses implicit self-adaptation, since the step length parameter is adapted through the evolutionary process itself.

In the following analysis, we show that there exists a function and initial conditions for which Algorithm A fails to converge to the globally optimal solution with probability one. Consider the following function:

$$g(x) = \begin{cases} |x| & , x < 10 \\ |x - 20| - 1 & , x \geq 10 \end{cases}.$$

This function is comprised of two local minima at $x = 0$ and $x = 20$. If $|x_0| < 10$ then this point is in the basin of attraction of the non-global local minima. Now if $\sigma_0 < \gamma(10 - |x_0|)$ then Algorithm A will begin searching within this basin of attraction (e.g. it cannot jump out in the first iteration). Locally, this function satisfies Assumption 1, and so we might expect that Algorithm A begins to search locally and thus misses the global optimum. The following theorem proves that this is true with some nonzero probability. Let $x^* = 20$ and recall that λ_1 is defined by Theorem 2.

Theorem 3 *Let $\{x_t\}$ be a sequence of points generated by Algorithm A on $g(x)$ starting with x_0 and σ_0 such that $|x_0| + W_\gamma \sigma_0 < 10$. If $\lambda \geq \lambda_1$ then $P(\lim_{t\to\infty} x_t = x^*) < 1$.*

Proof. Let $\overline{X}_t^\lambda$ and $\overline{\Sigma}_t^\lambda$ be the stochastic process, defined on some probability space $(\Omega, \mathcal{F}, P)$, that describes the behavior of Algorithm A on g. Consider the events in Ω for which Algorithm A converges to the global optimum: $A^* = \{\omega \in \Omega \mid \lim_{t\to\infty} \overline{X}_t^\lambda(\omega) = x^*\}$. We wish to show that $P(A^*) < 1$. Let $A_1 = \{\omega \in \Omega \mid \max_{t\geq 0} |\overline{X}_t^\lambda| \leq 10\}$, and note that $A_1^c \supseteq A^*$. Thus it suffices to show that $P(A_1) > 0$.

Now Z_t^λ is a non-negative super-martingale when Algorithm A is applied to the function $g'(x) = |x|$. Thus we can apply Kolgomorov's Theorem [8] to show that

$$P\left(\max_{t\geq 0} Z_t^\lambda > 10\right) < \frac{E(Z_0^\lambda)}{10} = \frac{|x_0| + W_\gamma \sigma_0}{10},$$

and from our assumptions we have $P(\max_{t\geq 0} Z_t^\lambda > 10) < 1 - \delta$ for some $\delta > 0$. Now consider the process $\overline{Z}_t^\lambda = |\overline{X}_t^\lambda| + W_\gamma \overline{\Sigma}_t^\lambda$ on the set of events $A_2 = \{\omega \in \Omega \mid \max_{t\geq 0} \overline{Z}_t^\lambda \leq 10\}$. For each event $\omega \in A_2$ the behavior of $Z_t^\lambda(\omega)$ is identical to $\overline{Z}_t^\lambda$. Thus we have $P(A_2) > \delta$. To conclude, note that $P(A_1) > P(A_2) > \delta > 0$. ∎

This result provides further theoretical evidence that we should not expect self-adaptive EAs to robustly perform global optimization for multimodal objective functions. This result complements Rudolph's result by considering a different form of self-adaptation. Additionally, our result applies to a *non-elitist* self-adaptive EA, and thus our result answers one of the open questions posed by Rudolph [14].

6 DISCUSSION

Rudolph [14] summarizes theoretical results concerning self-adaptive EAs and notes that the theoretical underpinnings for these methods are essentially unex-

plored. Our analysis exactly characterizes the stochastic process that underlies a class of self-adaptive $(1, \lambda)$-ESs, and we have developed a convergence theory that provides a robust guarantee that these methods converge. Our analysis is similar in spirit to Rudolph's analysis of non-elitist EAs [11]. The main difference in our work is the focus on EAs that *implicitly* self-adapt the mutation step length by coevolving these parameters within the EAs evolutionary process; our analysis appears to be the first result to exactly prove convergence for these self-adaptive EAs.

We expect that it will be difficult to prove similar results for general, nonconvex objective functions. However, we conjecture that you can relax the symmetry assumption that is made in our analysis. This assumption does not appear to be too critical for our proof that Z_t^λ is a super-martingale. However, this assumption may be more central to our proofs of Theorems 1 and 2, and this assumption is heavily exploited in our analysis of feasible values of ν_i and γ. Similarly, it should also be relatively straightforward to extend our analysis to the self-adaptive (μ, λ)-ES, which generates λ points and keeps the best μ for the next iterations. However, this may also weaken our analysis of the feasible values of ν_i and γ. In both of these cases, the more general convergence theory makes it more difficult to theoretically confirm that the convergence theory applies for small values of λ.

The use of discrete random variables in our self-adaptive EA is the key element that facilitates our analysis. This 'discrete' model of EAs on continuous design spaces is similar in spirit with our previous analyses of evolutionary pattern search methods [9, 10], where discretization of the mutation steps provides significant theoretical leverage. Although discrete self-adaptive EAs are not commonly used when optimizing continuous search domains, our empirical results suggest that these EAs may perform as well as standard self-adaptive EAs.

We believe that our result in Section 5 is quite interesting in several ways. First, this result illustrates the value of the underlying convergence theory that we have developed; our analysis is much simpler than Rudolph's analysis of explicitly self-adaptive EAs [14]. Second, this result confirms that despite their success as global optimizers, the self-adaptation in EAs limits their ability to perform global search. Thus as Rudolph notes, we should begin to focus on the global aspects of these methods. Finally, this result provides further justification for our analysis of evolutionary pattern search methods [9, 10], which focus on proving a local convegence theory. If we cannot expect that adaptation will provide global convergence, then we should ensure that it *does* provide local convergence on a wide range of objective functions.

We conclude by noting several open problems that are related to this work. First, it would be interesting to consider the progress rate for Algorithm A. We imagine that this analysis might follow directly from Beyer's analysis of the $(1, \lambda)$-ES. However, it may be possible exploit our use of discrete random variables to avoid some of the approximations that are made in that analysis

Next, the basic proof technique should be extensible to multidimensional problems. However, a significant complication is that in more than one dimension you can take a step that both increases the value of the objective function *and* the step length. Thus it appears that the basic convergence result for Z_t^λ may not be possible without further assumptions on the objective function.

Finally, a similar analysis should be possible for the self-adaptive $(1+\lambda)$-ES, which only replaces x_t in iteration t if one of the λ points generated is an improving point. This is an elitist EA, and consequently the discretized mutation steps would make it possible for this EA to get stuck near an optimum. Thus it is necessary to augment the EA to contract the step length with some fixed, nonzero probability in order to prevent it from converging to a point that is not locally optimal.

Acknowledgements

This work was supported by the DOE Office of Science, and it was performed at Sandia National Laboratories. Sandia is a multiprogram laboratory operated by Sandia corporation, a Lockheed Martin Company, for the United States Department of Energy under Contract DE-AC04-94AL85000.

References

[1] A. Agapie, *Theoretical analysis of mutation-adaptive evolutionary algorithms*, Evolutionary Computation, (2001), pp. 127–146.

[2] T. Bäck and H.-P. Schwefel, *An overview of evolutionary algorithms for parameter optimization*, Evolutionary Computation, 1 (1993), pp. 1–23.

[3] H.-G. Beyer, *Toward a Theory of Evolution Strategies: Self-Adaptation*, Evolutionary Computation, 3 (1995), pp. 311–347.

[4] ——, *The Theory of Evolution Strategies*, Springer-Verlag, 2001.

[5] J. D. DeLaurentis, L. Ferguson, and W. E. Hart, *On the convergence of an implicitly self-adaptive evolutionary algorithm: Symmetric, unimodal problems*, (2002). (submitted).

[6] A. E. Eiben, R. Hinterding, and Z. Michalewicz, *Parameter control in evolutionary algorithms*, IEEE Trans Evolutionary Computation, 3 (1999), pp. 124 – 141.

[7] G. W. Greenwood and Q. J. Zhu, *Convergence in evolutionary programs with self-adaptation*, Evolutionary Computation, (2001), pp. 147–158.

[8] G. R. Grimmett and D. R. Stirzaker, *Probability and Random Processes, Second Edition*, Oxford University Press, Oxford, 1992.

[9] W. E. Hart, *A convergence analysis of unconstrained and bound constrained evolutionary pattern search*, Evolutionary Computation, 9 (2001), pp. 1–23.

[10] ——, *Evolutionary pattern search algorithms for unconstrained and linearly constrained optimization*, IEEE Trans Evolutionary Computation, 5 (2001), pp. 388–397.

[11] G. Rudolph, *Convergence of non-elitist strategies*, in Proc of the First IEEE Conf on Evolutionary Computation, vol. 1, Piscataway, NJ, 1994, IEEE Press, pp. 63–66.

[12] ——, *Convergence Properties of Evolutionary Algorithms*, Kovač, 1997.

[13] ——, *Convergence rates of evolutionary algorithms for a class of convex objective functions*, Control and Cybernetics, 26 (1997), pp. 375–390.

[14] ——, *Self-adaptive mutations may lead to premature convergence*, IEEE Trans Evolutionary Computation, 5 (2001), pp. 410–414.

[15] G. Yin, G. Rudolph, and H.-P. Schwefel, *Analyzing* $(1, \lambda)$ *evolution strategy via stochastic approximation methods*, Evolutionary Computation, 3 (1996), pp. 473–489.

An Analysis of the Role of Offspring Population Size in EAs

Thomas Jansen
Krasnow Institute
George Mason University
Fairfax, VA 22030
tjansen@gmu.edu

Kenneth De Jong
Krasnow Institute
George Mason University
Fairfax, VA 22030
kdejong@gmu.edu

Abstract

Evolutionary algorithms (EAs) are general stochastic search heuristics often used to solve complex optimization problems. Unfortunately, EA theory is still somewhat weak with respect to providing a deeper understanding of EAs and guidance for the practitioner. In this paper we improve this situation by extending existing theory on the well-known (1+1) EA to cover the $(1 + \lambda)$ EA, an EA that maintains an offspring population of size λ. Our goal is to understand how the value of λ affects expected optimization time. We compare the $(1 + \lambda)$ EA with the (1+1) EA and prove that on simple unimodal functions no improvements are obtained for $\lambda > 1$. By contrast there are more complex functions for which sensible values of λ can decrease the optimization time from exponential to a polynomial of small degree with overwhelming probability. These results shed light on the role of λ and provide some guidelines for the practitioner.

1 INTRODUCTION

Evolutionary algorithms (EAs) are a broad class of stochastic search heuristics that include genetic algorithms (Goldberg 1989), evolution strategies (Schwefel 1995), and evolutionary programming (Fogel 1995). While there are countless reports of successful applications, EA theory is often concerned with either only isolated aspects of the algorithm or extremely simplified versions. One example are investigations of the so-called (1+1) EA, a very simple EA that uses a parent population of size 1 in each generation to create a single offspring by bit-wise mutation and replaces the parent by its offspring if the fitness if the offspring is not inferior to that of its parent (see, for example, Rudolph (1997), Garnier, Kallel, and Schoenauer (1999), and Droste, Jansen, and Wegener (2002)).

It has been known for some time that a simple (1+1) EA is often at least as efficient as much more sophisticated EAs (Juels and Wattenberg 1995; Mitchell 1995). This motivates the search for situations when the (1+1) EA is outperformed by more sophisticated EAs. For example, if we increase the parent population size μ, i.e., a $(\mu + 1)$ EA and introduce multi-parent reproduction, there are problems for which the use of uniform or 1-point crossover can reduce the expected optimization time from exponential to a polynomial of small degree (Jansen and Wegener 2001).

In this paper we focus on the role of the offspring population size λ in $(1 + \lambda)$ EAs. In particular, we would like to know when it makes sense to have $\lambda > 1$, and if so, what are reasonable values for λ. Our intuition suggests that, for simple unimodal functions, simple hill-climbing optimization procedures such as a (1+1) EA are efficient and not likely to be outperformed by $(1 + \lambda)$ EAs. However, as the complexity of the functions to be optimized increases, there should be benefits for having $\lambda > 1$.

We formally address these issues by extending the analysis of the (1+1) EA as described in Droste, Jansen, and Wegener (2002). In particular, we are able to characterize the slowdown due increasing λ on two well-known unimodal test functions: OneMax and LeadingOnes. In addition, we exhibit a class of functions for which increasing λ in a quite sensible manner reduces the expected optimization time from e^n to n^2, where n is the dimension of the search space. The result of this analysis is an improved understanding of the role of offspring population size and a better sense of how to choose λ for practical applications.

In the next section, we give a formal description of the $(1 + \lambda)$ EA and describe the two well-known test

functions investigated. In Section 3 we present results on the expected optimization time of the $(1+\lambda)$ EA for these test problems. In Section 4 we present an example function that allows us to see when the use of an offspring population can reduce the optimization time from exponential to a polynomial of small degree. However, modifying this example function we see that also the opposite behavior can be observed, i. e., the use of a quite small population increases the optimization from $O(n^2)$ to exponential. In Section 5, we conclude and describe possible future research.

2 DEFINITIONS

The $(1+\lambda)$ EA to be analyzed is an evolutionary algorithm that maintains a parent population of a size one to produce in each generation λ offspring independently and replaces the parent by a best offspring if the fitness of this offspring is not inferior to the fitness of its parent. The internal representation is a fixed length binary string of length n and offspring variation is the result of bit-wise mutation. For convenience we assume the optimization goal is maximization. More formally, we wish to maximize $f\colon \{0,1\}^n \to \mathbb{R}$ using:

Algorithm 1 ($(1+\lambda)$ EA).

1. Choose $x \in \{0,1\}^n$ uniformly at random.
2. For $i := 1$ To λ Do
   Create y_i by flipping each bit in x independently with probability $1/n$.
3. $m := \max\{f(y_1), \ldots, f(y_\lambda)\}$.
4. If $m \geq f(x)$ Then
   Replace x by one y_i chosen uniformly at random from $\{y_i \mid (1 \leq i \leq \lambda) \wedge (f(y_i) = m)\}$.
5. Continue at line 2.

Clearly, by setting $\lambda := 1$, we get the well-known (1+1) EA. Our goal is to analyze the effects that $\lambda > 1$ has on optimization performance. As usual in theoretical analysis we measure the optimization time in terms of the number of function evaluations needed before x becomes a globally optimal point for the first time. If G is the number of generations, i. e., the number of times lines 2–4 are executed, before this happens, we have $T = 1 + \lambda \cdot G$ for the optimization time T. Note that T is a fairer measure than the number of generations G. However, on a parallel machine a speed-up of the factor λ can be achieved. Since T obviously is a random variable, we are mostly concerned with $\mathrm{E}(T)$, the expected optimization time. Of course, other properties of T can be of interest, too.

3 PERFORMANCE ON SIMPLE UNIMODAL FUNCTIONS

Theoretical analyses of evolutionary algorithms often begin with simply structured example problems. The perhaps best known binary string test problem is OneMax, where the function value is defined to be the number of ones in the bitstring. Another slightly less trivial example is LeadingOnes, where the function value is given by the number of consecutive ones counting from left to right.

One way of analyzing algorithm performance on test problems is via "growth curve analysis" in which closed form expressions are sought that express the amount of time required to solve a problem as a function of the "size" of the problem (Corman, Leiserson, Rivest, and Stein 2001). For example, it is known that the expected optimization time $\mathrm{E}(T)$ of the (1+1) EA on OneMax is $\Theta(n \log n)$, i.e., as the size of the binary search space $\{0,1\}^n$ increases, the increase in $\mathrm{E}(T)$ is well-approximated by $n \log n$, and on LeadingOnes the expected optimization time is $\Theta(n^2)$ (Droste, Jansen, and Wegener 2002).

Both of these functions are examples of unimodal functions that can be optimized efficiently by simple hill-climbing algorithms such as a (1+1) EA. As a consequence, intuitively, one shouldn't expect a $(1+\lambda)$ EA to outperform a $(1+1)$ EA on such functions. Rather, one might expect a decrease in performance as we increase λ. In this section we show this more formally by extending the (1+1) EA growth curve analysis to $(1+\lambda)$ EAs. Interestingly, doing this for LeadingOnes is far simpler than for OneMax.

3.1 PERFORMANCE ON LEADINGONES

Theorem 1.1. *For the $(1+\lambda)$ EA on the function* LeadingOnes$\colon \{0,1\}^n \to \mathbb{R}$, $E(T) = \Theta(n^2 + n \cdot \lambda)$ *holds if λ is bounded above by a polynomial in n.*

Proof. We begin with the upper bound and distinguish two different cases with respect to the number of offspring λ. First, we assume that $\lambda \leq en$ holds. Note, that we want to prove that the expected optimization time is $\Theta(n^2)$. We begin with the upper bound. Obviously, it is sufficient if the current function value is increased by at least 1 at least n times in order to reach the unique global optimum 1^n. During the optimization the probability to create an offspring with larger function value is bounded below by $(1/n) \cdot (1-1/n)^{n-1} \geq 1/(en)$, since it is always sufficient to mutate exactly the left-most bit with value 0. The probability that in one generation no offspring

has larger function value than x is therefore bounded above by $(1-1/(en))^{\lambda}$. Thus, with probability at least $1-(1-1/(en))^{\lambda} \geq 1-e^{-\lambda/(en)}$ the current string x is replaced by one of its offspring with larger function value. Since we have $\lambda \leq e \cdot n$, obviously $\lambda/(en) \leq 1$ holds. For $t \in \mathbb{R}$ with $0 \leq t \leq 1$ we have $1-e^{-t} \geq t/2$. Thus, the probability that in one generation at least one offspring improves upon its parent x is bounded below by $\lambda/(2en)$ for $\lambda \leq en$. Therefore, the expected number of generations for an improvement of the function value is bounded above by $2en/\lambda$. We see, that the expected number of generations the $(1+\lambda)$ EA needs to optimize LeadingOnes is bounded above by $2en^2/\lambda$. So, the expected optimization time is at most $\lambda \cdot 2en^2/\lambda = O\left(n^2\right)$ as claimed.

Now, assume that $\lambda > en$ holds. As we noticed above, the probability that in one generation the $(1+\lambda)$ EA creates an offspring with larger function value than its parent is bounded below by $1-e^{-\lambda/en}$. Since we assume $\lambda > en$, this is bounded below by $1-e^{-1}$. Thus, the expected number of generations until we have n such generations is bounded above by $O(n)$ implying $O(\lambda \cdot n)$ as upper bound on $\mathrm{E}(T)$.

In order to derive a lower bound, we remember that λ is bounded above by some polynomial. So, there exists a constant k, such that $\lambda \leq n^k$ holds. We see that the probability of producing an offspring where $k+2$ pre-specified bits are mutated is bounded above by $1/n^{k+2}$. Thus, the probability that such an individual is not created among $n \cdot \lambda$ offspring is bounded below by $1-\left(1/n^{k+2}\right) \cdot (n \cdot \lambda) > 1-1/n$. Now, we consider only the first $n/(4k+8) = \Theta(n)$ generations. There at most $n^{k+1}/(4k+8)$ offspring are produced. Thus, with probability at least $1-1/n$ no such individual is created. Now, we work under the assumption that this is the case in the observed run. With probability $1/2$, at least $n/2$ bits in the initial string are 0. It is known (Droste, Jansen, and Wegener 2002), that the bits that are to the right of the leftmost bit with value 0 are random and independent during the run. Thus, with probability at least $(1/2)-1/n$, after $n/12$ generations the global optimum is not reached. This implies $\Omega(\lambda n)$ as a lower bound on $\mathrm{E}(T)$. □

This theorem confirms our intuition that a $(1+\lambda)$ EA will not outperform a $(1+1)$ EA on LeadingOnes since no value of $\lambda > 1$ will reduce the expected optimization time below $\Theta(n^2)$. However, the more interesting implication of this theorem is that, from a growth curve analysis point of view, there is no significant slowdown if λ is anywhere in the range $1 \leq \lambda \leq n$, since $\Theta\left(n^2+n\cdot\lambda\right) = \Theta\left(n^2\right)$ for $\lambda = O(n)$.

This is due to the fact that, in each step of the LeadingOnes function, the probability of creating an offspring that is better than its parent is $\Theta(1/n)$ (i.e., in ES terminology, the mutation operator has an almost constant success probability of $\Theta(1/n)$). Thus, producing up to n offspring in each generation before checking for improvements results in no substantial waste of time.

3.2 PERFORMANCE ON ONEMAX

For OneMax things are less obvious. Although it is fairly straightforward to show that a $(1+\lambda)$ EA cannot do better than a $(1+1)$ EA, it is considerably more difficult to pin down precisely when increasing λ begins to significantly affect performance. This is due to the fact that the success probability of mutation for OneMax varies over time rather than remaining almost constant as in LeadingOnes. As we approach the global optimum of 1^n, the probability of creating an offspring that is better than its parents approaches $\Theta(1/n)$. Hence, as we saw with LeadingOnes, any $\lambda \leq n$ is acceptable.

However, at the beginning, starting with a randomly generated binary string, the probability of success is with overwhelming probability larger than $1/c$ for a constant c, much larger than $1/n$. Hence, unless λ is a constant independent of n, there will be at least an initial slowdown. Consequently, we see that there is no simple intuitive answer as to when an increase in λ degrades performance. But it is possible to bracket λ with useful upper and lower bounds. We begin with a theorem that establishes an upper bound on $\mathrm{E}(T)$ and indirectly sets a lower bound for λ:

Theorem 1.2. *For the $(1+\lambda)$ EA on the function* OneMax: $\{0,1\}^n \to \mathbb{R}$, $E(T) = O(n \log n + n\lambda)$ *holds.*

Proof. Assume that the current string x has Hamming distance d from the global optimum, i. e., $d = n - \textsc{OneMax}(x)$. The probability to create an offspring y with $\textsc{OneMax}(y) > \textsc{OneMax}(x)$ is bounded below by $d/n(1-1/n)^{n-1} \geq d/(en)$. Thus, the probability not to increase the function value of x in one generation is bounded above by $(1-d/(en))^{\lambda} \leq e^{-d\cdot\lambda/(en)}$. So, the probability to increase the function value of x in one generation is bounded below by $1-e^{-d\cdot\lambda/(en)} \geq 1-1/\left(1+d\cdot\lambda/(en)\right) = (d\cdot\lambda)/(en+d\cdot\lambda)$. Thus, $\mathrm{E}(T)$ is bounded above by

$$\lambda \cdot \sum_{d=1}^{n} \frac{en+d\lambda}{d\lambda} = \lambda\left(n + \frac{en}{\lambda}\sum_{d=1}^{n}\frac{1}{d}\right) = O(n\lambda + n\log n).$$

□

In particular, if λ is bounded above by $\log n$, i.e., $\lambda = O(\log n)$, then the expected optimization time is bounded above by $n \log n$, which is no worse than a $(1+1)$ EA on ONEMAX. Similarly, a useful lower bound on $\mathrm{E}(T)$ can be obtained that can be used to establish an upper bound on λ:

Theorem 1.3. *For the $(1+\lambda)$ EA on the function* ONEMAX: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Omega(\lambda \cdot n/\log n)$ *holds if λ is bounded above by a polynomial in n.*

Proof. The probability to create an offspring y by mutating x with $\mathrm{ONEMAX}(y) \geq \mathrm{ONEMAX}(x) + d$ is bounded above by

$$\binom{n}{d} \cdot \frac{1}{n^d} \leq \frac{1}{d!} < \left(\frac{e}{d}\right)^d$$

for all $x \in \{0,1\}^n$ and all $d \in \{1, 2, \ldots, n - \mathrm{ONEMAX}(x)\}$. Therefore, the probability to create one such y within $n \cdot \lambda$ independent tries is bounded above by $n \cdot \lambda \cdot (e/d)^d$. We conclude that the probability to create one such y within $n \cdot \lambda$ independent tries with $d \in \{\log(n), \ldots, n - \mathrm{ONEMAX}(x)\}$ is bounded above by $n \cdot \lambda \cdot (e/\log n)^{\log n} = e^{-\Omega(\log(n) \log\log n)}$. Thus, the probability that within n generations the Hamming distance to the optimum is not decreased by at least $\log(n)$ in one single generation is $1 - 2^{-\Omega(\log(n) \log\log n)}$. Since after random initialization the Hamming distance to the global optimum is at least $n/2$ with probability $1/2$, this implies $\Omega((n/\log n) \cdot \lambda)$ as a lower bound on $\mathrm{E}(T)$ as claimed. □

Hence, if we allow λ to increase faster than $\log^2 n$, we begin to see a significant slowdown relative to the $n \log n$ performance of a $(1+1)$ EA on ONEMAX. Combining this with the previous theorem we see that keeping $\lambda < \log n$ means we'll do no worse than a $(1+1)$ EA. This still leaves open the possibility that there are values of $\lambda \leq \log^2 n$ that result in performance improvements over a $(1+1)$ EA. However, as our intuition suggests, this is not the case. We show this formally by proving that, whenever λ grows more slowly than $\sqrt{n}/\log n$ (an upper bound much larger than $\log^2 n$), $\mathrm{E}(T)$ is bounded below by $\Omega(n \log n)$:

Theorem 1.4. *For the $(1+\lambda)$ EA on the function* ONEMAX: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Omega(n \log n)$ *holds if $\lambda = o(\sqrt{n}/\log n)$.*

Proof. It is easy to see that at some point of time the Hamming distance between the current string x and the global optimum is within $\{\lceil\sqrt{n}/2\rceil, \ldots, \lceil\sqrt{n}\rceil\}$ with probability very close to 1. We consider a run of the $(1+\lambda)$ EA after this point of time. Then, the probability to create an offspring y with $\mathrm{ONEMAX}(y) \geq \mathrm{ONEMAX}(x) + 3$ is bounded above by $\binom{\sqrt{n}}{3} \cdot 1/n^3 < 1/n^{3/2}$. Thus, the probability to create at least one such individual within $\lambda \cdot n \log n$ independent tries is bounded above by $1 - \left(1 - 1/n^{3/2}\right)^{\lambda \cdot n \log n} = O(\lambda \cdot \log n/\sqrt{n}) = o(1)$. Now, we continue under the assumption that no such individual is created. Then, the Hamming distance can only be decreased by 1 or 2 in each generation. Given that the current Hamming distance between x and the global optimum is d, the probability to create an offspring that decreases this Hamming distance is bounded above by $2d/n$. Thus, the probability to do so in one generation is bounded above by $1 - (1 - 2d/n)^\lambda < 4\lambda \cdot d/n$. Thus, the expected number of generations is bounded below by $\sum_{d=1}^{\sqrt{n}} n/(4\lambda \cdot d) = \Omega((n/\lambda) \cdot \log n)$. This implies $\mathrm{E}(T) = \Omega(n \log n)$. □

The previous 3 theorems collectively tell us that we can increase λ up to $\log n$ without a significant degradation in performance, but not beyond $\log^2 n$, leaving the interval between $\log n$ and $\log^2 n$ unresolved. The final two theorems of this section narrow that gap considerably by increasing the lower bound slightly and decreasing the upper bound to within a constant factor of the lower bound. We begin with the lower bound:

Theorem 1.5. *For the $(1+\lambda)$ EA on the function* ONEMAX: $\{0,1\}^n \to \mathbb{R}$, $E(T) = \Theta(n \log n)$ *holds if $\lambda \leq (\ln n) \cdot (\ln\ln n)/(2 \ln\ln\ln n)$.*

Proof. The lower bound follows from Theorem 1.4. For $\lambda = O(\log n)$ the upper bound follows from Theorem 1.2. So, we assume $\lambda \geq \ln n$ from now. In particular, we define $\gamma := \lambda/\ln n$ and have $\gamma \geq 1$.

We divide a run of the $(1+\lambda)$ EA into two disjoint phases and count the number of function evaluations separately for each phase. Let T_1 denote this number for the first phase and let T_2 denote this number for the second phase. The first phase starts after random initialization and continues as long as $\mathrm{ONEMAX}(x) \leq n - n/\ln\ln n$ holds for the current string x. The second phase starts immediately after the end of the first phase and ends when the global optimum is reached. Obviously, for the expected optimization time $\mathrm{E}(T)$ we have $\mathrm{E}(T) = \mathrm{E}(T_1) + \mathrm{E}(T_2)$ with these definitions.

In order to get an upper bound on $\mathrm{E}(T_2)$, we reconsider the proof of Theorem 1.2. We see that $\mathrm{E}(T_2)$ is bounded above by $\lambda \cdot \sum_{d=1}^{n/\ln\ln n} (en + d \cdot \lambda)/(d \cdot \lambda) = \lambda \cdot ((n/ln\ln n) + (en/\lambda) \cdot \sum_{d=1} n/\ln\ln n 1/d) = O(\lambda \cdot (n/\ln\ln n) + n\log n) = O(n \log n)$. Here we need $\lambda = O(\log(n)/\log\log n)$.

Now we consider the first phase. Since we have $\text{OneMax}(x) \leq n - n/\ln\ln n$, the probability to create an offspring y by mutation of x with $\text{OneMax}(y) \geq \text{OneMax}(x) + \gamma$ is bounded below by

$$\begin{aligned} & \binom{n/\ln\ln n}{\gamma} \cdot \left(\frac{1}{n}\right)^{\gamma} \cdot \left(1 - \frac{1}{n}\right)^{n-\gamma} \\ \geq & \left(\frac{n}{\gamma \cdot n \cdot \ln\ln n}\right)^{\gamma} \cdot \frac{1}{e} = e^{-(1+\gamma\ln\gamma+\gamma\ln\ln\ln n)}. \end{aligned}$$

We conclude that the probability to create at least one such individual y in one generation is bounded below by $1 - \left(1 - e^{-(1+\gamma\ln\gamma+\gamma\ln\ln\ln n)}\right)^{\lambda} \geq \lambda/((e\ln n)+\lambda) \geq 1/(e+1)$ since we have $\gamma \leq (\ln\ln n)/(2\ln\ln\ln n)$ by assumption. Obviously, after less than n/γ such generations the first phase ends. This implies $O(\lambda \cdot n/\gamma) = O(n\log n)$ as upper bound on $\mathrm{E}(T_1)$. □

We cannot prove that $(\ln n)(\ln\ln n)/(2\ln\ln\ln n)$ is a sharp upper bound in the sense that any value for λ larger than this implies that the expected optimization is worse than $n\log n$. However, we can show that it is true when λ grows asymptotically faster than that.

Theorem 1.6. *For the* $(1+\lambda)$ *EA on the function* $\text{OneMax}\colon \{0,1\}^n \to \mathbb{R}$, $E(T) = \omega(n\log n)$ *holds if* $\lambda = \omega\left((\ln n)(\ln\ln n)/\ln\ln\ln n\right)$.

Proof. We use a different proof strategy in order to derive this tighter lower bound. We derive an upper bound on the expected decreasement in the Hamming distance in one generation. Then we use this upper bound in order to prove that it is not likely that the Hamming distance is decreased by a certain amount in a pre-defined number of generations.

In analogy to the proof of Theorem 1.4 it is easy to see that at some point of time the Hamming distance between the current string x and the global optimum is within $\{\lceil n/(2e)\rceil, \ldots, \lceil n/e\rceil\}$ with probability very close to 1. From this point of time on the probability to create an offspring y with $\text{OneMax}(y) \geq \text{OneMax}(x) + d$ is bounded above by $\binom{n/e}{d} \cdot 1/n^d < 1/d^d$.

Consider g generations of the $(1+\lambda)$ EA. Let x be the current string before the first generation and let x' be the current string after the g-th generation. Let $D_{\lambda,g,x}$ denote the advance in this g generations by means of Hamming distance, i. e. $D_{\lambda,g,x} := \text{OneMax}(x') - \text{OneMax}(x)$. Obviously, $D_{\lambda,g,x}$ depends on x and we have $\mathrm{Prob}(D_{\lambda,g,x} \geq d) \geq \mathrm{Prob}(D_{\lambda,g,y} \geq d)$ for all $d \in \{0,1,\ldots,n\}$ and all $x, y \in \{0,1\}^n$ with $\text{OneMax}(x) \leq \text{OneMax}(y)$. Since the function value of the current string of the $(1+\lambda)$ EA can never decrease, $\mathrm{E}(D_{\lambda,g,x}) \leq g \cdot \mathrm{E}(D_{\lambda,1,x})$ holds for all λ, g and x. So, we concentrate on $\mathrm{E}(D_{\lambda,1,x})$ now.

Obviously, $D_{\lambda,1,x}$ is a random variable that depends on λ and the current string x at the beginning of the generation. However, it is clear that $\mathrm{Prob}(D_{\lambda,1,x} \geq d) < \lambda/d^d$ holds for all x with $\text{OneMax}(x) \leq n - \lceil n/e\rceil$. From now on, we always assume that $\text{OneMax}(x)$ is bounded above in this way.

We are interested in $\mathrm{E}(D_{\lambda,1,x})$. Since $D_{\lambda,1,x} \in \{0,1,\ldots,n\}$, we have $\mathrm{E}(D_{\lambda,1,x}) = \sum_{i=1}^{n} \mathrm{Prob}(D_{\lambda,1,x} \geq d)$. For $d < (3\ln\lambda)/\ln\ln\lambda$ we use the trivial estimation $\mathrm{Prob}(D_{\lambda,1,x} \geq d) \leq 1$. For d with $(3\ln\lambda)/\ln\ln\lambda \leq d < (\lambda\ln\lambda)/\ln\ln\lambda$ we have

$$\frac{\lambda}{d^d} \leq \frac{e^{\ln\lambda}}{e^{((3\ln\lambda)/(\ln\ln\lambda))\cdot((\ln\lambda)-\ln\ln\ln\lambda))}} < \frac{1}{\lambda}$$

and use the estimation $\mathrm{Prob}(D_{\lambda,1,x} \geq d) \leq 1/\lambda$. Finally, for $d \geq (\lambda\ln\lambda)/\ln\ln\lambda$ we have $\lambda/d^d \leq e^{\lambda}/e^{\lambda\ln\lambda} < e^{-\lambda}$ and use the estimation $\mathrm{Prob}(D_{\lambda,1,x} \geq d) < e^{-\lambda} < 1/n$. These three estimations yield $\mathrm{E}(D_{\lambda,1,x}) < 4\ln\lambda/\ln\ln\lambda$ for the expected advance in one generation.

Let G denote the number of generations the $(1+\lambda)$ EA needs for optimization of OneMax. Of course, $\mathrm{E}(G) \geq t \cdot \mathrm{Prob}(G \geq t)$ holds for all values of t. As we argued above, with probability at least $1/2$ at some point of time we have that some x with $\text{OneMax}(x) \in \{n - \lceil n/e\rceil, \ldots, n - \lceil n/(2e)\rceil\}$ as current string x. Thus, $\mathrm{Prob}(G \geq t) \geq \mathrm{Prob}(D_{\lambda,t,x} < n/e)$ holds, if x is some string with at least n/e zero bits. This yields $\mathrm{E}(G) \geq (t/2) \cdot \mathrm{Prob}(D_{\lambda,t,x} < n/e) = (t/2) \cdot (1 - \mathrm{Prob}(D_{\lambda,t,x} \geq n/e))$. By Markov inequation we have $\mathrm{E}(G) \geq (t/2) \cdot (1 - \mathrm{E}(D_{\lambda,t,x})/(n/e)) \geq (t/2) \cdot (1 - e \cdot t \cdot \mathrm{E}(D_{\lambda,1,x})/n)$. Together with our estimation for $\mathrm{E}(D_{\lambda,1,x})$ we have $\mathrm{E}(G) \geq (t/2) \cdot (1 - 4e \cdot t \cdot \ln\lambda/(n \cdot \ln\ln\lambda))$. We set $t := (n\ln\ln\lambda)/(8e\ln\lambda)$ and get $\mathrm{E}(G) \geq n\ln\ln\lambda/(32e\ln\lambda)$ which implies $\mathrm{E}(T) = \Omega(n\lambda\ln\ln\lambda/\ln\lambda)$ for the expected optimization time $\mathrm{E}(T)$, since $T = G \cdot \lambda$ holds. It is easy to see, that this implies $\mathrm{E}(T) = \omega(n\log n)$ for $\lambda = \omega((\ln n)(\ln\ln n)/\ln\ln\ln n)$ as claimed. □

3.3 SUMMARY

The theorems in this section provide a clear picture of the performance of a $(1+\lambda)$ EA on OneMax. It never outperforms a $(1+1)$ EA. It's performance is about the same if λ doesn't get much bigger than $(\log n)(\log\log n)/2\log\log\log n$, and performance degrades significantly after that.

4 $(1+\lambda)$ EA Performance On More Complex Functions

The previous section showed that for simple functions like OneMax and LeadingOnes increasing λ does not improve $\mathrm{E}(T)$ over $(1+1)$ EA performance. In this section we focus on cases where increasing λ does improve performance. Intuitively, a necessary condition for it to be useful to invest more effort in the random sampling in the neighborhood of the current population is that the fitness landscape is to some degree misleading in the sense that a $(1+1)$ EA is more likely to get trapped on a local peak.

In this section we show that the performance improvement due to increased values of λ can be tremendous. We illustrate this with an artificial function for which we can prove that increasing λ from 1 to n reduces $\mathrm{E}(T)$ from e^n to n^2 with a probability that converges to 1 extremely fast.

The intuition for this function is quite simple as illustrated in Fig. 1. We want it to consist of a narrow path that leads to the optimum. However, while following that path, the algorithm is confronted multiple branch points, each with the property that from it there are a variety of paths leading uphill, but only the steepest one leads to the global optimum. Hence, as we increase λ we increase the likelihood of picking the correct branch point path.

The formal definition of this function is more complicated than the intuition. To simplify it somewhat, we divide the definition into two parts. First, we define a function $f\colon \{0,1\}^n \to \mathbb{R}$ that realizes the main idea but assumes that the initial string is 0^n.

Definition 1.1. *For $n \in \mathbb{N}$ we define $k := \lfloor \sqrt{n} \rfloor$. We use $|x| = \textsc{OneMax}(x)$ and define the function $f\colon \{0,1\}^n \to \mathbb{R}$ for all $x \in \{0,1\}^n$ by*

$$f(x) := \begin{cases} & \text{if } x = y0^{n-ik}1^{(i-1)k} \text{ with} \\ (2i+3)n + |x| & 1 \le i \le k/2 \\ & \text{and } y \in \{0,1\}^k \\ & \text{if } x = 0^{n-j}1^j \text{ with} \\ (2i+4)n + |x| & i := \lceil j/k \rceil,\ 1 \le i \le k/2, \\ & \text{and } i \cdot k \neq j \\ 0 & \text{otherwise} \end{cases}$$

The core function f contains one main path $0^i 1^{n-i}$. There are about $\sqrt{n}/2$ points on this path that are of special interest. At these points it is not only beneficial to add another one in the right side. The function value is also increased by flipping any of the left most $\lfloor \sqrt{n} \rfloor$ bits. Thus, at these points there are a variety of uphill paths to chose among. One path of the form $0^i 1^{n-i}$ leads to the global optimum, while paths of the form $y0^{i'}1^{n-i'-\lfloor\sqrt{n}\rfloor}$ with $y \in \{0,1\}^{\lfloor\sqrt{n}\rfloor}$ all lead to local optima. Therefore we call these special points on the path branching points.

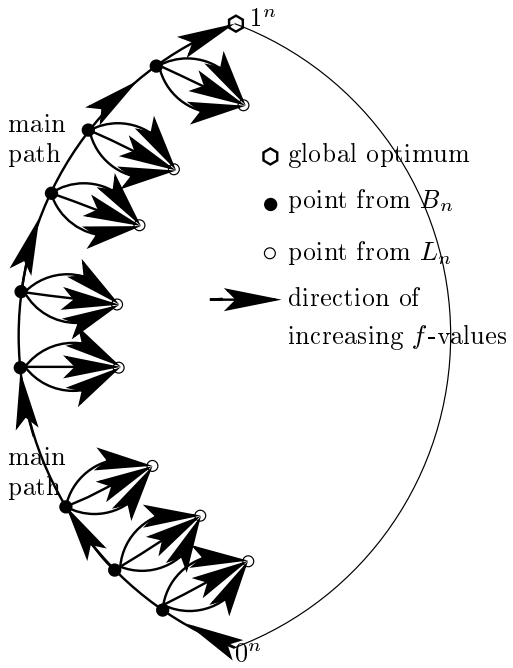


Figure 1: Core function f.

Definition 1.2. *For $n \in \mathbb{N}$ we define $k := \lfloor \sqrt{n} \rfloor$ and call a point $x \in \{0,1\}^n$ a* branching point of dimension n *iff $x = 0^{n-(i-1)k}1^{(i-1)k}$ holds for some $i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}$. Let B_n denote the set of all branching points of dimension n.*

At the branching points a $(1+\lambda)$ EA may proceed toward a local optima or the global one. It is important to know what the probabilities are for the two different possibilities:

Lemma 1.1. *Let $n \in \mathbb{N}$, $k := \lfloor \sqrt{n} \rfloor$, B_n the set of branching points of dimension n as defined in Definition 1.2, $x \in B_n$, and $\lambda\colon \mathbb{N} \to \mathbb{N}$ be a function that has a polynomial upper bound. Consider a $(1+\lambda)$ EA with current string x optimizing the function $f\colon \{0,1\}^n \to \mathbb{R}$ as defined in Definition 1.1. Let $y = y_1 y_2 \cdots y_n$ be the first point with $f(y) > f(x)$ reached by the $(1+\lambda)$ EA. The probability that $\textsc{OneMax}(y_1 y_2 \cdots y_k) > 0$ holds is $1 - e^{-\Theta(\lambda/\sqrt{n})}$.*

Proof. We consider the $(1+\lambda)$ EA on f with current string $x = x_1 x_2 \cdots x_n \in B_n$. Let $x' = x'_1 x'_2 \cdots x'_n$ be one individual created by mutation of x. We consider several events concerning x'.

The points in B_n are ordered according to the function value. The probability that x' has a function value that is equal or greater to the next branching point is obviously $2^{-\Omega(\sqrt{n})}$. We neglect such an event and take care of this in our upper and lower bounds by adding or subtracting $2^{-\Omega(\sqrt{n})}$.

We denote the event that $\textsc{OneMax}(x'_1 \cdots x'_k) > 0$ holds by A. We have

$$\text{Prob}(A) \geq \binom{k}{1} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} > \frac{k}{en}$$

as a lower bound and

$$\begin{aligned} & \text{Prob}(A) \\ \leq & \left(\sum_{i=1}^{k} \binom{k}{i} \left(\frac{1}{n}\right)^i \left(1 - \frac{1}{n}\right)^{n-i}\right) + 2^{-\Omega(\sqrt{n})} \\ < & \left(\sum_{i=1}^{k} \left(\frac{k}{in}\right)^i\right) + 2^{-\Omega(\sqrt{n})} < \frac{k}{n-k} \end{aligned}$$

for an upper bound. We denote the event that $\textsc{OneMax}(x'_1 \cdots x'_k) = 0$ holds by B and have

$$\text{Prob}(B) \geq \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-1} > \frac{1}{en}$$

as a lower bound and

$$\text{Prob}(B) \leq \left(\sum_{i=1}^{k} \left(\frac{1}{n}\right)^i \cdot \left(1 - \frac{1}{n}\right)^{n-i}\right) + 2^{-\Omega(\sqrt{n})} < \frac{1}{n}$$

for an upper bound.

We are interested in the probability of A given that A or B occur. Since $A \cap B = \emptyset$, we have $\text{Prob}(A|A \cup B) = \text{Prob}(A) / (\text{Prob}(A) + \text{Prob}(B))$. Obviously, for each $a, b, c \in \mathbb{R}^+$ we have that $a/(a+c) \geq b/(b+c)$ holds iff $a \geq b$. This implies that

$$\frac{k/(en)}{k/(en) + 1/n} < \text{Prob}(A|A \cup B) < \frac{k/(n-k)}{k/(n-k) + 1/(en)}$$

follows from our upper and lower bounds on $\text{Prob}(A)$ and $\text{Prob}(B)$ and we have $1 - e/\sqrt{n} < \text{Prob}(A|A \cup B) < 1 - 1/(e\sqrt{n})$. Let C denote the event that we are really interested in, the event that $\textsc{OneMax}(y_1 \cdots y_k) > 0$ holds for the first point $y = y_1 \cdots y_n$ with $f(y) > f(x)$ reached by the $(1+\lambda)$. Since in each generation the λ offspring are independently generated, we have $\text{Prob}(C) = 1 - (1 - \text{Prob}(A|A \cup B))^\lambda$ and can conclude that

$$1 - \left(\frac{e}{\sqrt{n}}\right)^\lambda < \text{Prob}(C) < 1 - \left(\frac{1}{e\sqrt{n}}\right)^\lambda$$

holds. Since $(1 - 1/x)^x < e^{-1} < (1 - 1/x)^{x-1}$ holds for all $x \in \mathbb{N}$, this completes the proof. □

Assume that the $(1+\lambda)$ EA happens to continue in the direction of a $y0^i1^{n-i-\lfloor\sqrt{n}\rfloor}$ with $y \in \{0,1\}^{\lfloor\sqrt{n}\rfloor}$, $y \neq 0^{\lfloor\sqrt{n}\rfloor}$. Then it is quite likely to reach "the last point in this direction", i. e., $y0^i1^{n-i-\lfloor\sqrt{n}\rfloor}$ with $y = 1^{\lfloor\sqrt{n}\rfloor}$. It will turn out be convenient to define a set for those points similar to B_n.

Definition 1.3. *For $n \in \mathbb{N}$ we define $k := \lfloor\sqrt{n}\rfloor$ and*

$$L_n := \left\{1^k 0^{n-ik} 1^{(i-1)k} \mid i \in \{1, 2, \ldots, \lfloor k/2 \rfloor - 1\}\right\}.$$

When considering the (1+1) EA on the core function f, it is essential that the algorithm is started with 0^n as initial string. Since the (1+1) EA choose the initial string uniformly at random this will not be the case with probability $1 - 2^{-n}$. Therefore, we now relax the assumption that the initial string is 0^n by extending f to g in a way that any starting point leads to the main path defined by f.

Definition 1.4. *For $n \in \mathbb{N}$ we define $m' := \lfloor n/2 \rfloor$, $m'' := \lceil n/2 \rceil$, and $g\colon \{0,1\}^n \to \mathbb{R}$ by*

$$g(x) := \begin{cases} n - \textsc{OneMax}(x'') & \text{if } x' \neq 0^{m'} \wedge x'' \neq 0^{m''} \\ 2n - \textsc{OneMax}(x') & \text{if } x' \neq 0^{m'} \wedge x'' = 0^{m''} \\ f(x'') & \text{if } x' = 0^{m'} \end{cases}$$

for all $x = x_1 x_2 \cdots x_n \in \{0,1\}^n$ with $x' := x_1 x_2 \cdots x_{m'} \in \{0,1\}^{m'}$ and $x'' := x_{m'+1} x_{m'+2} \cdots x_n \in \{0,1\}^{m''}$.

We double the length of the each bitstring and define the core function f only on the right half of the strings. The first half is used to lead a search algorithm towards the beginning of the main path of f. The construction will have the desired effect for all search heuristics that are efficient on $\textsc{OneMax}$.

Theorem 1.7. *The probability that the (1+1) EA optimizes the function $g\colon \{0,1\}^n \to \mathbb{R}$ within $n^{O(1)}$ steps is bounded above by $2^{-\Omega\left(\sqrt{n}\log n\right)}$.*

Proof. Our proof strategy is the following. First, we consider the expected optimization time of the (1+1) EA on the function $g\colon \{0,1\}^n \to \mathbb{R}$ under the condition that at some point of time the current string $x = x'x''$, with x' and x'' defined as in Definition 1.4, is of the form $x' = 0^{m'}$, $x'' \in L_{m''}$, with $m' = |x'|$, $m'' = |x''|$, $k = \left\lfloor\sqrt{m''}\right\rfloor$, and $i \in \{1, \ldots, \lfloor k/2 \rfloor\}$. Then, we prove that the probability that such a string becomes current string at some point of time is $\Omega(1)$.

First, assume that such a string x is current string of the (1+1) EA. Let A be the set of all such strings. Obviously, this string x is different from the unique global

optimum. Moreover, due to the definition of g, all points with larger function value have Hamming distance at least k and there are less than n such points. Thus, the probability to reach such a point in one generation is bounded above by $n \cdot (1/n)^k \leq (1/n)^{\sqrt{n/2}-2}$. The probability that such an event occurs at least once in n^k generations is bounded above by $n^{-\sqrt{n/2}+k+1} = 2^{-\Omega\left(\sqrt{n}\log n\right)}$.

The initial current string $x = x'x''$ after random initialization is some string with $x' \neq 0^{m'}$ with probability $1-2^{-m'}$. Then, the function value is either given by $n - \textsc{OneMax}(x'')$ or $2n - \textsc{OneMax}(x')$. With probability $1-2^{-\Omega(n)}$ the all zero string 0^n is reached within $O\left(n^2 \log n\right)$ steps given that no other string y with $g(y) > g\left(0^n\right)$ is reached before. There are less than $2^k \cdot n$ such strings, thus, the probability to reach such a string within $O\left(n^2 \log n\right)$ steps is bounded above by

$$O\left(n^2 \log n \cdot \frac{2^k \cdot n}{2^n}\right) = 2^{-n+O\left(\sqrt{n}\right)}.$$

At each branching point, the (1+1) EA comes to a string $x = x'x''$ as new current string with some bit set to 1 within the first k bits of x'' with probability at least $1 - 1/(e\sqrt{n})$. This follows from the proof of Lemma 1.1. The probability to proceed with such strings instead of returning to a string with k zero at these bit positions increases. In fact, it is easy to see that with probability $1 - O\left(1/\sqrt{n}\right)$ the $(1+\lambda)$ EA reaches a point in A before reaching some point with k zeros at these positions. Since we have $\lfloor k/2 \rfloor - 1 > \sqrt{n}/5$ branching points, we conclude that the probability that the (1+1) EA does not reach some point in A is bounded above by $2^{-\Omega\left(\sqrt{n}\log n\right)}$ under the described circumstances. Combining all estimations completes the proof. □

Theorem 1.8. *For all constants $\varepsilon > 0$, the probability that the $(1+\lambda)$ EA with $\lambda = n \cdot \lambda'$, $\lambda' \in \mathbb{N}$ optimizes the function $g\colon \{0,1\}^n \to \mathbb{R}$ within $O\left(n^2 \cdot \lambda'\right)$ steps is bounded below by $1-\varepsilon$.*

Proof. First, assume that the initial string $x = x'x''$ is some string with $x' \neq 0^{m'}$. Given that the all zero string is the first string $y = y'y''$ becoming current string with $y' = 0^{m'}$, we can conclude from Theorem 1.2 that the expected number of steps the $(1+\lambda)$ EA needs to reach the all zero string is $O\left(n^2 \cdot \lambda'\right)$. Similar to the proof of Theorem 1.7 we can conclude that the all zero string is reached within $cn^2 \cdot \lambda$ steps with probability at least $1-\varepsilon$, where c and ε are positive constants. Note, that by enlarging c the failure probability ε can be made arbitrarily small. Given that for all following current strings $x = x'x''$ we have that x'' does not contain a bit different from 0 within the first k bits, it follows from Theorem 1.1, that the expected number of steps until the global optimum is reached is bounded above by $O\left(n^2 \cdot \lambda'\right)$. So, under the assumption that no such string is reached, we have similarly to the reasoning above that the global optimum is reached within $O\left(n^2 \cdot \lambda'\right)$ steps with probability at least $1-\varepsilon$, where ε is an arbitrarily small positive constant. We know from Lemma 1.1 that the probability to reach some x where this assumption is not met is bounded above by $e^{-\Omega\left(\sqrt{n}\right)}$ at each branching point. Since there are less than $\sqrt{n}$ branching points, we have that with probability at least $1 - \sqrt{n} \cdot e^{-\Omega\left(\sqrt{n}\right)} = 1 - e^{-\Omega\left(\sqrt{n}\right)}$ no such point becomes current point x at any branching point. Combining these estimations completes the proof. □

At the same time, as has been seen in many other contexts, there is no "free lunch" here in the sense that the impressive speedup on g achieved by increasing λ is not true for all functions with local optima. For example, it is quite easy to modify the function of the previous section so that a (1+1) EA is efficient and a $(1+\lambda)$ EA with $\lambda \geq n$ fails with high probability. To see how, consider f and all strings $x = x'x''$ where x'' is a branching point. Let x''' be a string of length $|x''|$ with $x''' \in L_{|x'''|}$. that has k bits with the value 1 at the beginning and is equal to x'' on the rest of the bits. The proofs of Theorem 1.7 and Theorem 1.8 rely on the fact the the (1+1) EA reaches some string $x'x'''$ with high probability whereas the $(1+\lambda)$ EA does not. Defining a function f' that has all such point $x'x'''$ as global optimum and is equal to f on all other points is obviously optimized within $O\left(n^2\right)$ steps by the (1+1) EA with probability converging to 1, whereas the $(1+\lambda)$ fails to optimize f' within a polynomial number of steps with probability converging to 1, given that $\lambda = \Omega(n)$ holds.

5 CONCLUSIONS

We have used growth curve analysis techniques to better understand the role of offspring population size in $(1+\lambda)$ EAs. As we have seen, increasing λ on simple functions like OneMax and LeadingOnes does not lead to an improvement in the expected optimization time. However, increases in λ do not significantly degrade performance as long as $\lambda < \log n$ where n is the dimensionality of the search space. By contrast, for more complex functions with local optima, increasing λ can result in improvements in performance as illustrated in the previous section.

However, a note of caution needs to be raised. The growth curve analysis techniques used here focus on limiting behavior as n increases and view growth curves that differ only by a constant as equivalent. Hence, it may be the case that specific values of n and λ produce "contradictory" results to the conclusions in the previous paragraph.

In spite of this cautionary note we believe that these insights into the role of offspring population size provide some guidelines for the EA practitioner. Since practitioners apply EAs to functions about which they do not have detailed *a priori* knowledge, they can "hedge their bets" by choosing λ to be of order $\log n$. In doing so, they don't lose too much if the function turns out to be easy and may improve their performance on functions with local optima. An interesting observation here is that for most practical problems, $\log n$ is not likely to be too far away from the value of $\lambda = 7$ recommended by the ES community but derived in other contexts (Schwefel 1995, p. 148). On the other hand, if the function appears to be quite difficult in the sense that each run results in convergence to a different local optima, increasing λ to a value of order n is a plausible thing to try.

This work represents a modest step along to road to a more general EA theory. It extends some of the techniques and results known for the $(1 + 1)$ EA to $(1 + \lambda)$ EAs. Clearly, additional steps are required, including the extension of these results to $(\mu + \lambda)$ EAs. From a more practical perspective, the heuristic guidelines proposed here for choosing values for λ need to be experimentally evaluated and tied more closely to properties of functions.

Acknowledgments

The first author was supported by a fellowship within the post-doctoral program of the German Academic Exchange Service (DAAD).

References

T. Corman, C. Leiserson, R. Rivest, and C. Stein (2001). *Introduction to Algotrithms, Second Edition*. New York, NY: McGraw Hill.

S. Droste, T. Jansen, and I. Wegener (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*. Accepted for publication.

D. B. Fogel (1995). *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence*. Piscataway, NJ: IEEE Press.

J. Garnier, L. Kallel, and M. Schoenauer (1999). Rigorous hitting times for binary mutations. *Evolutionary Computation 7*(2), 173–203.

D. E. Goldberg (1989). *Genetic Algorithms in Search, Optimization, and Machine Learning*. Reading, MA: Addison-Wesley.

T. Jansen and I. Wegener (2001). Real royal road functions — where crossover provably is essential. In L. Spector, E. D. Goodman, A. Wu, W. B. Langdon, H.-M. Voigt, M. Gen, S. Sen, M. Dorigo, S. Pezeshk, M. H. Garzon, and E. Burke (Eds.), *Proceedings of the Genetic and Evolutionary Computation Conference* (*GECCO 2001*), San Francisco, CA, 1034–1042. Morgan Kaufmann.

A. Juels and M. Wattenberg (1995). Hillclimbing as a baseline method for the evaluation of stochastic optimization algorithms. In D. S. Touretzky (Ed.), *Advances in Neural Information Processing Systems* (*NIPS 8*), Cambridge, MA, 430–436. MIT Press.

M. Mitchell (1995). *An Introduction to Genetic Algorithms*. MIT Press.

G. Rudolph (1997). *Convergence Properties of Evolutionary Algorithms*. Hamburg, Germany: Verlag Dr. Kovač.

H.-P. Schwefel (1995). *Evolution and Optimum Seeking*. New-York, NY: Wiley.

On the Dynamics of Evolutionary Multi-Objective Optimisation

Tatsuya Okabe, Yaochu Jin, Bernhard Sendhoff
HONDA R&D Europe (Deutschland) GmbH
Future Technology Research
Carl-Legien-Strasse 30, 63073 Offenbach/M, Germany
{tatsuya.okabe, yaochu.jin, bernhard.sendhoff}@de.hrdeu.com

Abstract

In evolutionary multi-objective optimisation (EMOO) using dynamic weighted aggregation (DWA), very interesting dynamic behaviours of the individuals have been observed [7] [8]. In this paper, the dynamics of the individuals on fitness space (FS) during multi-objective optimisation (MOO) using evolution strategies (ES) is studied by investigating the mapping of a normal distribution in the parameter space (PS) onto the FS. It is found that the movement of the individuals on the FS is strongly dependent on the characteristics of the projected distribution. Results on three test functions are given to show the good agreement of dynamics predicted from theoretical calculation with that observed in MOO using DWA.

1 INTRODUCTION

Evolutionary algorithms (EA) have been shown to be very successful for multi-objective optimisation (MOO) problems. Up to now a variety of methods for MOO have been proposed [2, 3]. In addition, theoretical studies on the accuracy of the approximation of the Pareto front, on the convergence properties and on the diversity of individuals in a population have also been reported. However, the dynamics of individuals during optimisation, that is, the characteristics of the movement of the individuals on the fitness space (FS) has not yet been investigated to the best of our knowledge. This might be attributed to the fact that in our notion the fitness space[1] of single objective optimisation problems is one dimensional and it can be argued that we might not learn much from observing the dynamics of the population on one axis.

However, in MOO the situation is different, the space is at least two-dimensional and it is believed that the investigation of the dynamics of individuals on the FS will lead to a deeper insight into the properties of Pareto fronts and to a better understanding of the working mechanism of MOO algorithms, which will ultimately enable us to improve the performance of MOO algorithms.

In this paper, we will study the dynamics of the individuals on the FS empirically and analytically by concentrating on the mapping of the mutation distribution from PS to the FS. This approach is motivated by the way evolution strategies produce offsprings, i.e., by adding normally distributed random vectors to the parent parameter vector. Therefore, by understanding the changes the fitness function induces on the normal distribution, we can understand some of the properties of multi-objective evolutionary algorithms based on evolution strategies. At the same time, the notion of a search distribution is not restricted to evolution strategies and has been proposed as a unified approach to the analysis of evolutionary algorithms, see e.g. [10]. Thus, we are confident that the general approach presented in this paper is not restricted to evolution strategies.

The work in this paper is partly motivated by the behaviours observed in MOO using the dynamically weighted aggregation (DWA) algorithm [7, 8]. The basic idea is to combine the optimisation objectives with different weights, which are changed dynamically during optimisation so that a set of Pareto-optimal solutions instead of one single solution will be obtained. It has been shown that the method is not only effective for problems with a convex Pareto front, but also for those with a concave Pareto front. In the opti-

[1] This should not be confused with the notion of fitness landscape, where the fitness values are mapped over the parameter or more general genotype/phenotype configurations.

misation, it is found that when the weights change, the individuals move along the Pareto front once they reach one point on it, even if the Pareto front is discontinuous. To understand why individuals follow the Pareto front was the initial target of this work. Note, that this type of movement is even observed for sudden large weight changes and *after* mutation but *before* selection. Therefore, the trivial explanation that the individuals simply follow the Pareto front since it is the path of "highest fitness" is not sufficient.

Of course, we had to choose objective functions for which we carry out our analysis, we chose three functions (concave, convex, discontinuous). In the course of our work, it turned out that the presented analysis can also be used to determine whether and when (e.g. with respect to the initialisation) a test problem is difficult or not for a specific algorithm; we will comment on these findings in Section 5.

The remainder of this paper is structured as follows: In the next section, we will briefly recall some of the theoretical work on MOO. In Section 3, we will concisely outline Evolution Strategy, the DWA method, the test functions and present some of the empirical observations. In Section 4, we will analyse the transformation of the mutation distribution and relate it to the results from Section 3. Further implications of Section 4, e.g. for the difficulty of test functions are discussed in Section 5.

2 THEORY FOR EMOO

Results on the convergence of evolutionary multiobjective optimisation have been presented by Rudolph [12, 13] based on the Markov chain approach which has been successfully used for the analysis of single objective evolutionary algorithms, see e.g. [11] among others. The work by Hanne [6] is also mainly concerned with the convergence of evolutionary multiobjective algorithms. Complexity issues have been addressed for example by van Veldhuizen [17]. Very recently an interesting approach has been suggested by Thiele et al. [16] to define a simple problem class for multi-objective optimisation to facilitate the theoretical analysis of evolutionary algorithms for this domain.

Teich presented some theoretical investigations for uncertain objectives for MOO [15], based on which he developed the Estimate Strength Pareto Evolutionary Algorithm (ESPEA).

Since the comparison of different algorithms for multiobjective optimisation is much harder than for single objective ones, it has also been the focus of some theoretical investigations, which are mainly based on a statistical approach using an appropriate metric, see e.g. the work by Fonseca et al. [5] and by Zitzler et al. [19].

Even though the above list is likely to be incomplete, compared to the overall number of publications in MOO, theoretical approaches have been sparse in particular for the analysis of the dynamics of individuals during the evolutionary search and of the main factors which determine the characteristics of this movement. Surely the approach in this work can only be seen as a starting point, however, we believe it can be beneficial to explain some of the empirical observations, which we will outline in the next section.

3 DYNAMICALLY WEIGHTED AGGREGATION

3.1 EVOLUTION STRATEGIES

In evolution strategies (ES), mutation plays the major role in search. The mutation is performed by adding a random number generated from a normal distribution $N(0, \sigma_i^2)$, where σ_i is the standard deviation. In the standard ES, new individuals are generated in the following way [1]:

$$\vec{x}(t) = \vec{x}(t-1) + \vec{z} \quad (1)$$

$$\sigma_i(t) = \sigma_i(t-1)\exp(\tau' z)\exp(\tau z_i), \quad (2)$$

where, $\vec{x}$ is an n-dimensional parameter vector, $\vec{z}$ is an n-dimensional random number vector with $\vec{z} \sim N(0, \sigma(t)^2)$, z and z_i are normally distributed random numbers with $z, z_i \sim N(0,1)$. In ES, the σ_i are also called stepsizes, and are evolved together with the object parameters. This is known as self-adaptation, which is an important feature of the ES.

The parameters τ and τ' in equation (2) are constants that are given as follows:

$$\tau = \frac{1}{\sqrt{2\sqrt{n}}} \quad (3)$$

$$\tau' = \frac{1}{\sqrt{2n}} \quad (4)$$

In the ES usually a deterministic selection method is used. In the non-elitist (μ, λ) method, the best μ individuals from the λ offspring are chosen as the parents of the next generation.

3.2 BASIC IDEA OF DWA

Jin et al. [7, 8] proposed dynamically weighted aggregation as an efficient method to easily apply any evolu-

tionary algorithm (and evolution strategies in particular) to multi-objective optimisation problems. Since our empirical observations are based on the DWA method, we will explain it briefly in the following.

The basic idea is to linearly combine all objectives like in the conventional aggregation method: $f = \sum_{i=1}^{m} w_i f_i$. Here m, w_i and f_i are the number of objective functions, the weights for the f_i and the objective functions, with $\sum_{i=1}^{m} w_i = 1$. In order to obtain the whole Pareto front, the weights w_i are changed dynamically in each generation using a periodical function between $[0, 1]$, for example, the sine function. To achieve the whole Pareto front, it is necessary to maintain an archive of non-dominated solutions.

Whereas conventional weighted aggregation methods with restart cannot obtain concave Pareto fronts, it has been argued and empirically demonstrated in [8] that the DWA methods indeed can (at least if some mild assumptions about the changing functions for the $w_i(t)$ are made).

3.3 TEST FUNCTIONS

Several test functions for multi-objective optimisation have been proposed in the literature, see e.g. [9, 18, 4, 14]. Here we chose three functions whose Pareto front can be calculated analytically and which represent the three important cases of convex, concave and discontinuous Pareto fronts.

3.3.1 Function T_1 (Convex Case)

The first test function T_1 is defined as follows [9]:

$$f_1 = \frac{1}{n}\sum_{i=1}^{n} x_i^2 \tag{5}$$

$$f_2 = \frac{1}{n}\sum_{i=1}^{n} (x_i - 2.0)^2 \tag{6}$$

The convex Pareto front can be calculated analytically with the following result:

$$f_2 = f_1 - 4\sqrt{f_1} + 4 \tag{7}$$
$$0 \le f_1 \le 4, 0 \le f_2 \le 4$$

3.3.2 Function T_2 (Discontinuous Case)

The second test function T_2 is defined as follows [18]:

$$f_1 = x_1 \tag{8}$$

$$f_2 = g \times \left(1 - \sqrt{\frac{f_1}{g}} - \frac{f_1}{g}\sin(10\pi f_1)\right) \tag{9}$$

$$g(x_2, \ldots, x_n) = 1 + \frac{9}{n-1}\sum_{i=2}^{n} x_i$$
$$x_i \in [0, 1]$$

The discontinuous Pareto front is:

$$f_2 = 1.0 - \sqrt{f_1} - f_1 \sin(10\pi f_1), \tag{10}$$

where f_1 can be from the following intervals: $f_1 \in [0.0000, 0.0830]$, $(0.1823, 0.2579]$, $(0.4095, 0.4541]$, $(0.6187, 0.6528]$, $(0.8237, 0.8523]$. These constraints on f_1 were not obtained analytically, but from simulations. As it is evident, the Pareto front is discontinuous.

3.3.3 Function T_3 (Concave Case)

The third test function T_3 is defined as follows [4, 14]:

$$f_1 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i - \frac{1}{\sqrt{n}}\right)^2\right) \tag{11}$$

$$f_2 = 1 - \exp\left(-\sum_{i=1}^{n}\left(x_i + \frac{1}{\sqrt{n}}\right)^2\right) \tag{12}$$

This original test function ($n = 8$) was proposed by Fonseca and Fleming in 1993. Here we generalised it to the n-dimensional case. The concave Pareto front is given by:

$$f_2 = 1 - \frac{1 - f_1}{e^4}\exp\left(4\sqrt{-\log(1 - f_1)}\right) \tag{13}$$
$$f_1 = [0, 1 - e^{-4}]$$

3.4 DYNAMICS OF THE DWA ALGORITHM

When we apply the DWA algorithm to the three test functions described in the previous section and observe the dynamics of the individuals in particular during the movement along the Pareto front, some interesting phenomena can be observed. Unfortunately, we cannot *show* the dynamics directly, therefore, we have to present snapshots for different generations and describe the behaviour in between.

For all experiments, the following weight change function was used:

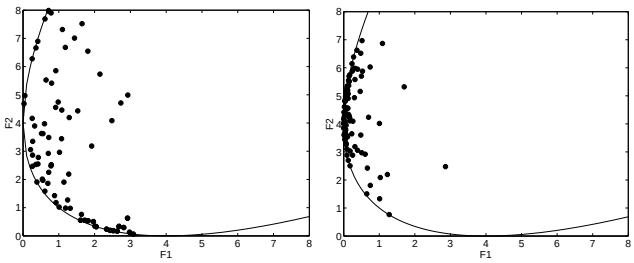
$$w_1 = \frac{1}{2}\text{sign}\left(-\sin(0.1\pi\, t)\right) + \frac{1}{2}, \tag{14}$$

where t is the number of generations.

In this paper, we use standard ES with $(\mu, \lambda) = (15, 100)$, and the archive size is 200. The initial standard deviation is 0.1, 0.01 and 0.1 for test functions

T_1, T_2 and T_3 respectively. Two dimensional cases, i.e. $n = 2$, are shown.

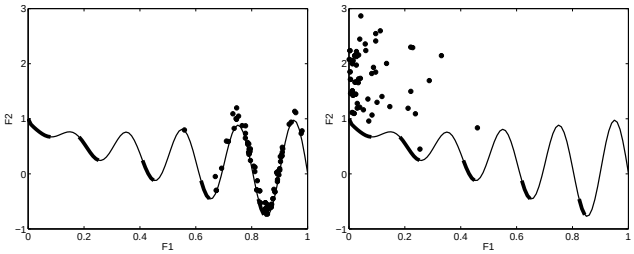
In Figure 1 the distributions of individuals (circles) *after* mutation and *before* selection[2] are shown for test function T_1. Whereas in Figure 1(a) the distribution is fairly widespread, in Figure 1(b) is concentrated on the Pareto front. In both cases selection has little influence on the shape of the individual's distribution which is mainly determined by the shape of the mutation distribution of the parents. Even though, one can argue that the diversity is considerably decreased in later parent generations, this does not account for the non-isotropic nature of the distribution, which is particularly evident from Figure 1 (b), where nearly all offspring - before selection - are located on the Pareto front. Indeed if we observe the continuous dynamics, we see that the individuals move nearly perfectly along the Pareto front, which makes the search very efficient.



(a) (b)

Figure 1: Distribution of the individuals *after* mutation and *before* selection for test function T_1 for the DWA algorithm after generation 1 in figure (a) and generation 18 in figure (b).

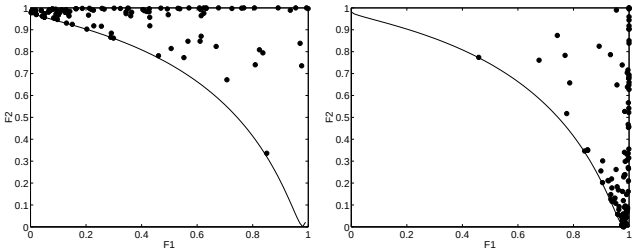
Similar dynamics can be observed for the second test function T_2, for which snapshots of the distribution of the individuals before selection are shown in Figure 2. The solution to equation (10) is given by the thin curve and the Pareto front (all non-dominated solutions), which consists of parts of this curve, is given by the thicker curve elements. Again we see that when the parents are located near $(f_1, f_2) = (1, 0)$ at generation 18, (Figure 2(a)), the individuals are restricted to the set of non-dominated solutions, whereas near $(f_1, f_2) = (0, 1)$ at generation 29, (Figure 2(b)), the distribution of individuals is rather wide spread. Indeed, dynamically one can nicely observe, how the individuals move from $(f_1, f_2) = (0, 1)$ to $(f_1, f_2) = (1, 0)$ and back along the thin curve being wider distributed near the left end and strongly concentrated near the right end of the curve in Figure 2. This movement can be observed several times during the periodic change of the weights, according to equation (14).



(a) (b)

Figure 2: Distribution of the individuals *after* mutation and *before* selection for test function T_2 for the DWA algorithm after generation 18 in figure (a) and generation 29 in figure (b).

For the third test function the results for generation 16 and 26 are shown in Figure 3(a) and (b). We observe that individuals are clustered near three points: $(f_1, f_2) = (0, 1)$, $(f_1, f_2) = (1, 0)$ and $(f_1, f_2) = (1, 1)$. The interior of this "triangle" bounded from below by the Pareto front, is very sparsely represented. During the dynamical observation, the peculiarity of the three points becomes even more evident, since in some generations nearly all individuals are concentrated in these points irrespective of the weight changes.



(a) (b)

Figure 3: Distribution of the individuals *after* mutation and *before* selection for test function T_3 for the DWA algorithm after generation 16 in figure (a) and generation 26 in figure (b).

We can summarise our observations as follows:

- The distribution of individuals strongly depends

[2] In each generation the complete evolution cycle (mutation *and* selection) is carried out. However, our snapshots are shown for one generation, say t_1, and the distribution is shown after mutation$_{t_1}$ and before selection$_{t_1}$ in order to highlight that the non-isotropic distribution of the individuals is to a large degree a result of mutation and not just of selection.

on the position in search space nearly irrespective of selection.

- The movement of individuals between points on the Pareto front follows a very distinctive pattern which is also not only controlled by selection.

4 THE SHAPE OF THE MUTATION DISTRIBUTION IN FITNESS SPACE

We have argued in the previous section that the characteristics of our empirical observations for all three test functions is to a large extent independent of selection. Therefore, it has to depend on the shape of the mutation distribution which is for evolution strategies in PS the normal distribution. Of course, when we observe the movement of individuals in FS, we must consider what the shape of the normal distribution looks like in FS. We will see that the transformed distribution can have very distinctive features which help to explain our empirical observations. The fitness values at generation t after mutation are given by

$$f(x(t)) = f(x(t-1)+z),\ z \sim N(0,\sigma_t^2) \tag{15}$$

Here f, $x(t)$ and σ_t are the objective function, the design variable at generation t, and the standard deviation at generation t. In this section, we neglect self-adaptation, which implies $\sigma_t = \sigma$. We will take it into consideration in the future work.

Without loss of generality, we restrict the following analytical investigation to the case $n = 2$.

In the two-dimensional case, the normal distribution on PS is given by:

$$f(x_1,x_2) = \frac{1}{2\pi\sigma_1\sigma_2} e^{-\frac{(x_1-\mu_1)^2}{2\sigma_1^2}} e^{-\frac{(x_2-\mu_2)^2}{2\sigma_2^2}} \tag{16}$$

Here, $f(x_1,x_2)$, σ and μ are the probability density function (pdf), the standard deviation and the mean, respectively.

Now, let us firstly assume that f_1 and f_2 are one-to-one functions, i.e. $x = f^{-1}(f(x))$. The other case, e.g. test function T_1, will be dealt with later. From this assumption, we can get the following equation:

$$\int_{f_1}^{f_1+\Delta f_1}\int_{f_2}^{f_2+\Delta f_2} g(f_1',f_2')df_2'df_1' = \int_{x_1}^{x_1+\Delta x_1}\int_{x_2}^{x_2+\Delta x_2} f(x_1',x_2')dx_2'dx_1' \tag{17}$$

Here $g(f_1,f_2)$ is the pdf in the FS. The area (x_1,x_2) - $(x_1+\Delta x_1, x_2+\Delta x_2)$ corresponds to (f_1,f_2) - $(f_1+\Delta f_1, f_2+\Delta f_2)$. From equation (17) we obtain

$$g(f_1,f_2) = \frac{1}{|J|} f(x_1,x_2), \tag{18}$$

where J is the Jacobian matrix.

Now we can calculate the mutation distribution in FS for each of the three test functions and compare the results with the observations of Section 3.

4.1 FUNCTION T_1 (CONVEX CASE)

$$g(f_1,f_2) = \frac{1}{2|x_2-x_1|}\{f(x_1,x_2)+f(x_2,x_1)\} \tag{19}$$

$$(x_1,x_2) = \frac{1}{4}(f_1-f_2+4)(1,1) + (\pm c, \mp c), \tag{20}$$

with c given by

$$c = \frac{1}{4}\sqrt{-f_1^2-f_2^2-16+2f_1f_2+8f_1+8f_2} \tag{21}$$

In order to obtain equation (19), we had to slightly modify equation (17) to take the fact into account that T_1 is not one-to-one:

$$\int_{f_1}^{f_1+\Delta f_1}\int_{f_2}^{f_2+\Delta f_2} g(f_1',f_2')df_2'df_1' = \int_{x_1}^{x_1+\Delta x_1}\int_{x_2}^{x_2+\Delta x_2} f(x_1',x_2')dx_2'dx_1' + \int_{-x_1-\Delta x_1}^{-x_1}\int_{-x_2-\Delta x_2}^{-x_2} f(x_1',x_2')dx_2'dx_1' \tag{22}$$

The results for some points in the FS are shown in Figure 4, the standard deviations are given by $(\sigma_1,\sigma_2) = (1,1)$.

Figures 4 demonstrate that once one solution on the Pareto front is found the individuals will move along the Pareto front with a high probability - independent of selection - solely because the shape of the normal distribution which defines mutation in evolution strategies is mapped to the shape shown in figures 4 (a)-(c). However, when the Pareto front has not been reached yet, i.e. the individual is concentrated in the interior, the search distribution is nearly isotropic and the success solely depends on selection, as shown in Figure 4(d).

4.2 FUNCTION T_2 (DISCONTINUOUS CASE)

$$g(f_1,f_2) = \frac{1}{9-\frac{9\sqrt{x_1}}{2\sqrt{1+9x_2}}} f(x_1,x_2) \tag{23}$$

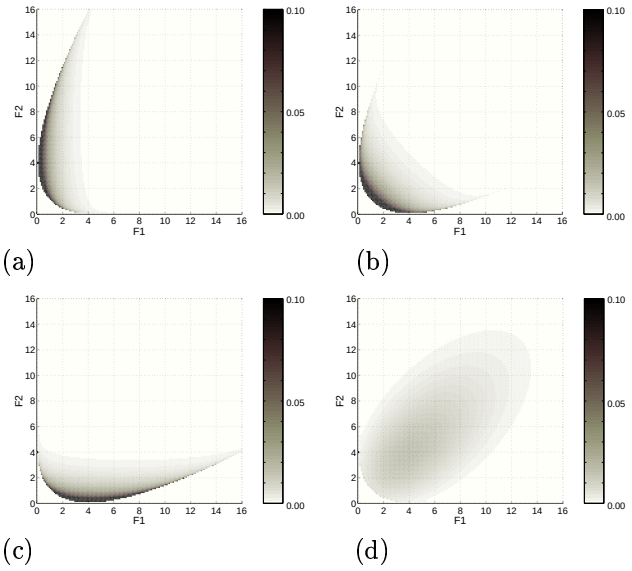


Figure 4: The shape of the normal distribution in FS for test function T_1. (a) $(x_1, x_2) = (0, 0)$, $(f_1, f_2) = (0, 4)$; (b) $(x_1, x_2) = (1, 1)$, $(f_1, f_2) = (1, 1)$; (c) $(x_1, x_2) = (2, 2)$, $(f_1, f_2) = (4, 0)$; (d) $(x_1, x_2) = (-1, 3)$, $(f_1, f_2) = (5, 5)$.

$$
\begin{aligned}
(x_1, x_2) &= \left(f_1, \frac{1}{9}(f_2 + \frac{f_1}{2} + f_1 \sin(10\pi f_1) + c) \right) \\
c &= \sqrt{f_1 f_2 + \frac{f_1^2}{4} + f_1^2 \sin(10\pi f_1)} - 1
\end{aligned}
$$

From the shape of the transformed mutation distributions in Figure 5 (a) and (b) we conjecture that individuals located at $(f_1, f_2) = (1, 0)$ are very likely to move along the curve of non-dominated solutions and discover the discontinuous Pareto front. Whereas, the opposite direction, i.e. the movement of individuals starting at $(f_1, f_2) = (0, 1)$ along this curve, is much less likely since the distribution is more isotropic.

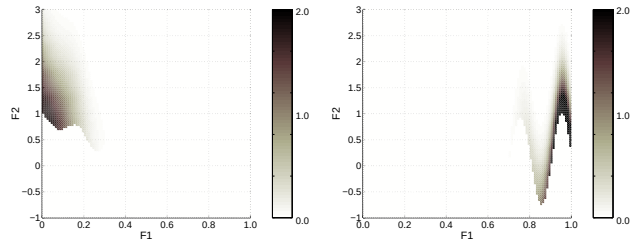


Figure 5: The shape of the normal distribution in FS for test function T_2. (a) $(x_1, x_2) = (0, 0)$, $(f_1, f_2) = (0, 1)$; (b) $(x_1, x_2) = (1, 0)$, $(f_1, f_2) = (1, 0)$.

4.3 FUNCTION T_3 (CONCAVE CASE)

$$
\begin{aligned}
g(f_1, f_2) &= \frac{1}{|J|} \{ f(x_1, x_2) + f(x_2, x_1) \} && (24) \\
J &= 4\sqrt{2} \exp\left(-2(x_1^2 + x_2^2 + 1) \right) (x_1 - x_2) \\
(x_1, x_2) &= \frac{\sqrt{2}}{8}(h_1 - h_2)(1, 1) + (\pm c, \mp c) && (25)
\end{aligned}
$$

Here, c, h_1 and h_2 are:

$$
\begin{aligned}
c &= \sqrt{-h_1^2 - h_2^2 - 16 + 2h_1 h_2 + 8h_1 + 8h_2} \\
(h_1, h_2) &= (-\log(1 - f_2), -\log(1 - f_1))
\end{aligned}
$$

Figure 6 shows the logarithm of the transformed mutation distribution in FS for different individuals positioned on the Pareto front as well as on the interior. We observe that in all cases the "boundaries" have a very high probability whereas the interior of the shown part of the fitness space has a very small probability. Thus, even without selective pressure individuals are very likely to move along the concave Pareto front, simply due to the shape of the transformed mutation distribution.

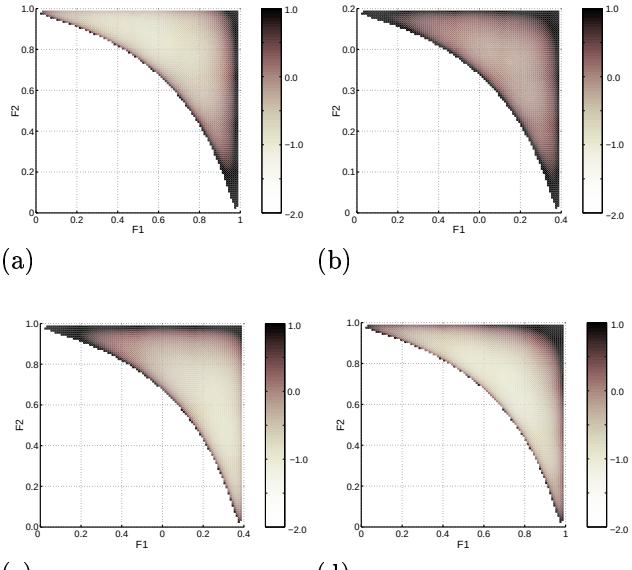


Figure 6: The shape of the normal distribution in FS for test function T_3, logarithmic values are shown. (a) $(x_1, x_2) = (-\frac{1}{\sqrt{2}}, -\frac{1}{\sqrt{2}})$, $(f_1, f_2) = (1 - e^{-4}, 0)$; (b) $(x_1, x_2) = (0, 0)$, $(f_1, f_2) = (1 - e^{-1}, 1 - e^{-1})$; (c) $(x_1, x_2) = (\frac{1}{\sqrt{2}}, \frac{1}{\sqrt{2}})$, $(f_1, f_2) = (0, 1 - e^{-4})$; (d) $(x_1, x_2) = (-1, 1)$, $(f_1, f_2) = (1 - e^{-3}, 1 - e^{-3})$;

5 FURTHER STUDIES ON FUNCTION T_1

In order to better understand the constraints which lead to the "compression" of the mutation distribution onto the Pareto front in some cases, we look at Function T_1 again in a bit more detail.

The Pareto front in the parameter and the fitness space for function T_1 is shown in Figure 7 by the thick curve, which is a straight line in PS. The parallel lines $x_2 = x_1 \pm c\sqrt{2}, (0 \leq x_1 + c\sqrt{2}/2 \leq 2)$, above and below the PS-Pareto line in a distance c are projected to the following curves:

$$f_2 = f_1 - 4\sqrt{f_1 - \frac{1}{2}c^2} + 4. \tag{26}$$

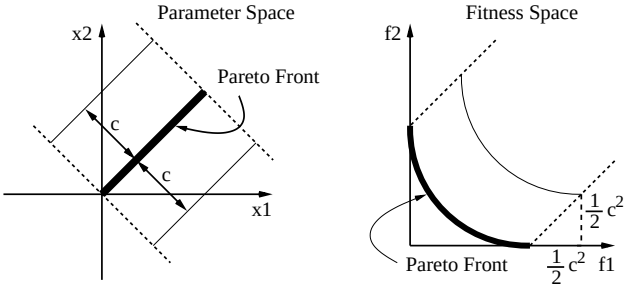
Equation (26) can be written as follows:

$$f_2 - \frac{1}{2}c^2 = f_1 - \frac{1}{2}c^2 - 4\sqrt{f_1 - \frac{1}{2}c^2} + 4, \tag{27}$$

with the constraints

$$0 \leq f_1 - \frac{1}{2}c^2 \leq 4 \tag{28}$$

$$0 \leq f_2 - \frac{1}{2}c^2 \leq 4. \tag{29}$$



(a) (b)

Figure 7: (a) The Pareto front in PS and parallel lines with distance c; (b) the Pareto front in FS and the images of the parallel lines in FS.

From the above considerations and from Figure 7, we can better understand in which way the mutation distribution is changed. The distance to the Pareto front in PS for individuals which lie on one of the parallel lines is c, in FS it is $\frac{1}{\sqrt{2}}c^2$. Therefore, for $c = \sqrt{2}$ this distance remains unchanged. Whereas for $c < \sqrt{2}$ the distance is decreased or "compressed" under the mapping of function T_1, it is increased for $c > \sqrt{2}$.

In the context of the probability distribution, it means that if the individual is located below the thin curve $\frac{1}{\sqrt{2}}c^2$ for $c = \sqrt{2}$ in Figure 7 (b), the probability of the area closer to the Pareto front is increased. The opposite holds for individuals above this curve for which it becomes more unlikely to move towards the Pareto front.

For function T_1 the recommended initialisation [9] of the parameters is $-2 \leq x_1 \leq 2$ and $-2 \leq x_2 \leq 2$. Therefore, the uniform probability density is given by:

$$p(x_1, x_2) = \begin{cases} 0.0625 & , -2 \leq x_i \leq 2 \quad (i = 1, 2) \\ 0 & , \text{else} \end{cases} \tag{30}$$

In fitness space this probability density is projected as follows:

$$\begin{aligned} p(f_1, f_2) &= \begin{cases} \frac{1}{8b} & , -2 \leq \frac{1}{4}\{a \pm b\} \leq 2 \\ 0 & , \text{else} \end{cases} \\ a &= f_1 - f_2 + 4 \\ b &= \sqrt{-f_1^2 - f_2^2 - 16 + 2f_1f_2 + 8f_1 + 8f_2} \end{aligned} \tag{31}$$

Equation (31) is shown in Figure 8 with logarithmic scale, it agrees well with simulations which we carried out.

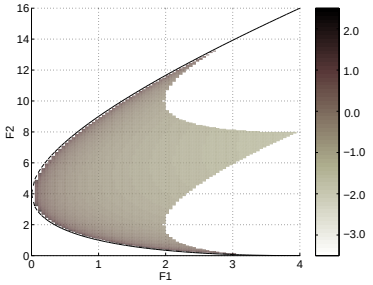
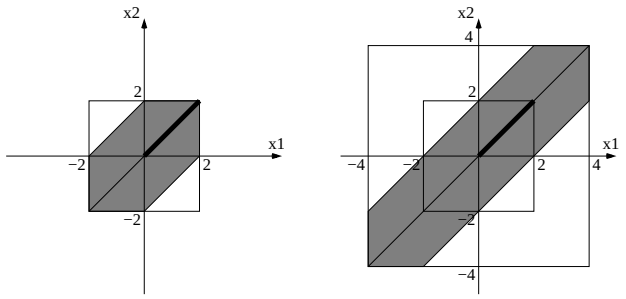


Figure 8: The shape of the probability density in fitness space for individuals, which have uniform distribution on $[-2, 2]$ in parameter space for function T_1.

We observe that the probability is very high for points on or close to the boundary including Pareto front and that it decreases with increased distance from the boundary. From our considerations above, this can be easily understood. For all points, which lie within the corridor show in Figure 7 for $c = \sqrt{2}$, their distance to the boundary (the Pareto front is the section of the boundary between the two coordinates) is reduced under the mapping. For the square $-2 \leq x_i \leq 2$, $(i = 1, 2)$, these are $3/4$ of all points, thus the probability density is increased near the boundary, see Figure 9.

In Figure 9 the thick lines represent the Pareto front on PS and the gray areas are "compressed" regions.



(a) (b)

Figure 9: The area of initialisation on T_1. (a) shows the initialisation $[-2, 2]$. The gray ("compressed") area covers 3/4 of the total region. (b) shows $[-4, 4]$. The gray area covers 7/16 of the total region.

We conclude that the initialisation of the individuals, even if it is uniform in parameter space, might result in an important bias in fitness space. In particular for problem T_1 we can conclude that the proposed initialisation will lead to an initial population where it is rather likely that some individuals are already located on or very near the Pareto front. If this initialisation is e.g. used together with the DWA method the performance of the algorithm will be very high, however, mostly because of the specific relations between PS and FS. Generally, we would advise to use an initialisation $-4 \leq x_i \leq 4$, $(i = 1, 2)$, at least, where the number of points whose distance is decreased is roughly equal to the ones whose distance is increased (relation 7/9) to reduce the probability that initial individuals are already located near the Pareto front. The shape of the probability density in the fitness space is shown in Figure 10 when the parameters are initialised on $[-4, 4]$ in parameter space.

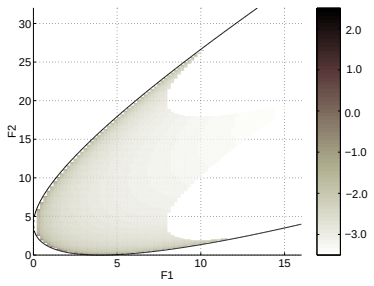


Figure 10: The shape of the probability density in fitness space for individuals, which have uniform distribution on $[-4, 4]$ in parameter space for function T_1.

6 CONCLUSION

In this paper, we investigated the dependence of the dynamics of the individuals in fitness space on the properties of the mapping of the probability density function for mutation or more general for the population of the next generation from parameter space to fitness space. The analysis which we presented here is restricted to evolution strategies or at least to those evolutionary algorithms where a normally distributed mutation is the main operator, e.g. evolutionary programming. Although we did not explicitly test this, we are very confident that the results of this paper are valid for any selection method, indeed the analytical investigations in Section 4 and 5 are completely independent from selection.

We believe our approach can be a starting point for a more general investigation of the influence of the PS-FS mapping on the search distribution which is usually only discussed in the PS. In particular for multi-objective optimisation where FS is at least two-dimensional, the movement of the individuals on this space can show a much richer dynamics. Although the importance of the PS-FS mapping on all aspects of evolutionary algorithms is widely accepted, the analysis of its influence on the search without selection has not received much attention so far. The fact that in some cases the Pareto front is a local attractor for the population (in a probabilistic sense, see Figure 4) without the influence of selection seems to be worth noticing.

We started our analysis with the observation of some dynamic behaviour of the DWA method under some conditions. However, as we have shown in the former section, the results are not restricted to the DWA or related methods, but bear strong implications on such important questions, like "When is a test function difficult?". In the literature this question is usually discussed in the context of such properties like deceptiveness, ruggedness, etc., however, the much simpler notion of the relation of distances in PS and in FS is hardly addressed. Nevertheless, as Figure 8 and 10 shows, it might have a very direct influence on the algorithm's performance, without telling much about the true *strength* of the algorithm.

Acknowledgements

The authors would like to thank E. Körner for his support and T. Arima and M. Olhofer for discussions on the subject.

References

[1] T. Bäck. *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[2] C.A. Coello Coello. A Comprehensive Survey of Evolutionary-Based Multiobjective Optimization Techniques. *Knowledge and Information Systems*, 1(3):269–308, 1999.

[3] K. Deb. *Multi-Objective Optimization Using Evolutionary Algorithms*. John Wiley & Sons, 2001.

[4] C.M. Fonseca and P.J. Fleming. Genetic algorithm for multiobjective optimization, formulation, discussion and generalization. In *Genetic Algorithms: Proceedings of the Fifth International Conference*, pages 416–423, 1993.

[5] C.M. Fonseca and P.J. Fleming. On the Performance Assessment and Comparison of Stochastic Multiobjective Optimizers. In *Lecture Notes in Computer Science 1141 Parallel Problem Solving from Nature - PPSN IV*, pages 584–593, 1996.

[6] T. Hanne. On the convergence of multiobjective evolutionary algorithms. *European Journal of Operational Research*, 117(3):553–564, 1999.

[7] Y. Jin, T. Okabe, and B. Sendhoff. Adapting Weighted Aggregation for Multiobjective Evolution Strategies. In *Lecture Notes in Computer Science 1993 Evolutionary Multi-Criterion Optimization*, pages 96–110, 2001.

[8] Y. Jin, M. Olhofer, and B. Sendhoff. Dynamic Weighted Aggregation for Evolutionary Multi-Objective Optimization: Why Does It Work and How? In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1042–1049, 2001.

[9] J.D. Knowles and D.W. Corne. Approximating the Nondominated Front Using the Pareto Archived Evolution Strategy. *Evolutionary Computation*, 8(2):149–172, 2000.

[10] H. Mühlenbein and T. Mahnig. Evolutionary Algorithms: From Recombination to Search Distributions. In *Theoretical Aspects of Evolutionary Computing*, pages 137–176. Springer, 2000.

[11] G. Rudolph. Convergence Analysis of Canonical Genetic Algorithms. *IEEE Transactions on Neural Networks*, 5(1):96–101, 1994.

[12] G. Rudolph. On a Multi-Objective Evolutionary Algorithm and Its Convergence to the Pareto Set. *Proceedings of the 5th IEEE Conference on Evolutionary Computation*, pages 511–516, 1998.

[13] G. Rudolph and A. Agapie. Convergence Properties of Some Multi-Objective Evolutionary Algorithms. In *Congress on Evolutionary Computation 2000*, volume 2, pages 1010–1016, 2000.

[14] K.C. Tan, E.F. Khor, C.M. Heng, and T.H. Lee. Exploratory Multi-objective Evolutionary Algorithm: Performance Study and Comparisons. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pages 647–654, 2001.

[15] J. Teich. Pareto-Front Exploration with Uncertain Objectives. In *Lecture Notes in Computer Science 1993 Evolutionary Multi-Criterion Optimization*, pages 314–328, 2001.

[16] L. Thiele, K. Deb, and E. Zitzler. Talk presented at the Dagstuhl Seminar "Theory of Evolutionary Computation", January 2002.

[17] D.A. van Veldhuizen and G.B. Lamont. Multiobjective Evolutionary Algorithms: Analyzing the State-of-the-Art. *Evolutionary Computation*, 8(2):125–147, 2000.

[18] E. Zitzler, K. Deb, and L. Thiele. Comparison of Multiobjective Evolutionary Algorithms: Empirical Results. *Evolutionary Computation*, 8(2):173–195, 2000.

[19] E. Zitzler and L. Thiele. Multiobjective Evolutionary Algorithms: A Comparative Case Study and the Strength Pareto Approach. *IEEE Transactions on Evolutionary Computation*, 3(4):257–271, 1999.