UNIVERSITY OF DORTMUND

REIHE COMPUTATIONAL INTELLIGENCE

COLLABORATIVE RESEARCH CENTER 531

Design and Management of Complex Technical Processes and Systems by means of Computational Intelligence Methods

Methods for the Analysis of Evolutionary Algorithms on Pseudo-Boolean Functions

Ingo Wegener

No. CI-99/00

Technical Report ISSN 1433-3325 September 2000

Secretary of the SFB 531 · University of Dortmund · Dept. of Computer Science/XI
44221 Dortmund · Germany

This work is a product of the Collaborative Research Center 531, "Computational Intelligence", at the University of Dortmund and was printed with financial support of the Deutsche Forschungsgemeinschaft.

METHODS FOR THE ANALYSIS OF EVOLUTIONARY ALGORITHMS ON PSEUDO-BOOLEAN FUNCTIONS*

Ingo Wegener
FB Informatik, LS2, Univ. Dortmund, 44221 Dortmund, Germany
wegener@ls2.cs.uni-dortmund.de

Abstract Many experiments have shown that evolutionary algorithms are useful randomized search heuristics for optimization problems. In order to learn more about the reasons for their efficiency and in order to obtain proven results on evolutionary algorithms it is necessary to develop a theory of evolutionary algorithms. Such a theory is still in its infancy. A major part of a theory is the analysis of different variants of evolutionary algorithms on selected functions. Several results of this kind have been obtained during the last years. Here important analytical tools are presented, discussed, and applied to well-chosen example functions.

1. INTRODUCTION

Evolutionary algorithms are randomized search heuristics with many applications, e.g., in optimization, adaptation, classification, control systems, or learning. Here we focus on optimization (for an overview on the whole area we refer to Bäck, Fogel, and Michalewicz (1997), Fogel (1995), Goldberg (1989), Holland (1975), and Schwefel (1995)). Despite the many successful experiments with evolutionary algorithms a theory on evolutionary algorithms is still in its infancy. This holds in particular if one compares the state of the art with the situation on problem-specific deterministic exact optimization algorithms (Cormen, Leiserson, and Rivest (1990)), deterministic approximation algorithms (Hochbaum (1997)), or randomized optimization and approximation algorithms (Motwani and Raghavan (1995)). One reason is that evolutionary algorithms have been developped by engineers, while the other

*This work was supported by the Deutsche Forschungsgemeinschaft (DFG) as part of the Collaborative Research Center "Computational Intelligence" (SFB 531).

disciplines have been created by theoreticians (leading sometimes to a lack of experimental knowledge). Moreover, the fundamental idea of evolutionary algorithms is to obtain robust problem-independent search heuristics with a good behavior on many problems from a large variety of problems (this statement remains true, although many evolutionary algorithms also use problem-specific components). This variety of problems makes the analysis of evolutionary algorithms much harder than the analysis of problem-specific algorithms (which often are designed in order to make an analysis possible). Nevertheless, progress on the design and the application of evolutionary algorithms will gain a lot from a theoretical foundation. Nowadays, we are able to analyze evolutionary algorithms without crossover on many functions and evolutionary algorithms with crossover on some functions. The functions are not examples from real-world applications but example functions describing some typical issues of functions (fitness landscapes) or are chosen to show some extreme behavior of evolutionary algorithms. Also very strange functions can be useful in order to disprove widely accepted conjectures or to show the differences between different variants of evolutionary algorithms. Altogether, we find a list of interesting theoretical results on evolutionary algorithms, in particular, during the last years. The purpose of this contribution is to present some of the most important tools for such results.

In Section 2, we discuss differences between discrete and non-discrete state spaces and why we investigate the optimization of pseudo-boolean functions $f : \{0,1\}^n \rightarrow \mathbb{R}$. The aim of an analysis of an evolutionary algorithm is the investigation of certain performance measures. This paper focusses on expected run times and the success probability within reasonable time bounds. The reasons for this decision are presented in Section 3. Since several example functions are used for different purposes all these functions are defined in Section 4. The following three sections show how tail inequalities (Section 5), the coupon collector's theorem (Section 6), and results on the gambler's ruin problem (Section 7) can be applied to the analysis of evolutionary algorithms. Another main idea is to measure the progress of an evolutionary algorithm not with respect to the considered fitness function but with some cruder scale. In Section 8 and Section 9, upper and lower bounds on the expected run time of evolutionary algorithms are proved using levels based on intervals of fitness values. The method of using so-called potential functions for the analysis of algorithms is well established. We discuss the first successful application of this powerful tool for evolutionary algorithm in Section 10. Finally, in Section 11, we present the method of designing "typical runs" of an evolutionary algorithm. Then we can bound the time of a typical run and can estimate the failure probability, i.e., the probability that a run is not typical.

2. OPTIMIZATION OF PSEUDO-BOOLEAN FUNCTIONS

Although many of our methods and results can be transferred to the non-discrete case, we focus here on the somehow simpler or clearer case of discrete functions. We also abstract from possible constraints which leads to the search space $S = \{0,1\}^n$.

Definition 1 *Functions* $f : \{0,1\}^n \to \mathbb{R}$ *are called pseudo-boolean.*

Without loss of generality we investigate the maximization of pseudo-boolean fitness functions. While technical systems lead to non-discrete search spaces, the whole area of combinatorial optimization leads to our scenario. Pseudo-boolean functions always have optimal search points. Search heuristics evaluate each point of the search space in expected finite time implying that expected run times are finite. Moreover, the search space has some nice features. The minimal Hamming distance between different points is 1 while the maximal distance is n. Hence, evolutionary algorithms with fixed mutation probabilities or fixed expected length of mutation steps can optimize pseudo-boolean functions efficiently. For fitness functions on $\mathbb{R}^n$ and even on some compact subspace of $\mathbb{R}^n$ it is necessary to decrease the expected length of steps in order to approximate the optimum. Nevertheless, the chance of meeting the optimum exactly is often 0.

In order to present the techniques to analyze evolutionary algorithms for simple examples we often investigate the perhaps simplest randomized search heuristic belonging to the class of evolutionary algorithms.

Algorithm 1 (1+1)EA

- *Choose* $x \in \{0,1\}^n$ *randomly.*
- *Let* x' *be the result of mutating* x*, i.e., the bits of* x'_i *are generated independently and* $x'_i = \overline{x}_i$ *with the mutation probability* $1/n$ *and* $x'_i = x_i$ *otherwise.*
- *Replace* x *with* x' *iff* $f(x') \geq f(x)$.
- *Repeat the last two steps until a stopping criterion is fulfilled.*

Often we consider the Markoff process describing the randomized search with the (1+1)EA as infinite process without stopping rule. Then we are interested in the first point of time where something nice happens, e.g., the first point of time where a global optimal search point is evaluated. The investigation of the (1+1)EA is less limited as one may believe. We may use the multi-start option, i.e., we consider p independent runs of the (1+1)EA. This is often better or at least as good as the consideration of a population of size p. The independency of runs ensures the diversity. For larger populations, one has to ensure the diversity by certain tricks. A population size of 1 does not admit crossover. However, the analysis of evolutionary algorithms without crossover is difficult enough and we consider evolutionary algorithms with crossover only in Section 11. The mutation probability of $1/n$ is the most often recommended choice (Bäck (1993)). Alternating mutation probabilities by self-adaptation have been investigated by Beyer (1996) and Bäck (1998) and a dynamic variant of the (1+1)EA has been analyzed by Jansen and Wegener (2000a).

3. PERFORMANCE MEASURES

The analysis of an algorithm is the task to describe quantitatively the behavior of the algorithm. Let X_f be the random variable measuring the first point of time when some event happens. The most natural choice of such an event is that an optimal search point is evaluated. However, one may also consider the case that some search point whose fitness is sufficiently close to the fitness of an optimal point or the case that some search point is sufficiently close to an optimal point. The run time analysis of a randomized algorithm consists of

- the computation or estimation of $E(X_f)$,
- the analysis of the so-called success probability distribution $Pr(X_f \leq t)$, and
- the analysis of the best case, worst case, or average case of $E(X_f)$ and $Pr(X_f \leq t)$ with respect to functions f from some class F of functions.

The investigation of the success probability includes the investigation of multi-start variants. E.g., if $Pr(X_f \leq t) \geq \frac{1}{n}, n$ independent runs have after t steps a success probability of at least $1-(1-\frac{1}{n})^n \geq 1-e^{-1}, n \log n$ independent runs improve the success probability to $1-O(\frac{1}{n})$ and n^2 runs even to $1-2^{-\Omega(n)}$. There are examples where $E(X_f)$ grows exponentially while the success probability after a polynomial number of steps is bounded below by a positive constant.

Expected run time and success probability are global performance measures and are those performance measures which typically are used for randomized algorithms (see Motwani and Raghavan (1995)). In the theory of evolutionary algorithms many other aspects are considered. These other performance measures are of certain value, but we think that their analysis finally is only a tool to get results on the global behavior. In order to understand the global behavior it is useful to understand the local behavior. Quality gain and progress rate (see Bäck, Fogel, and Michalewicz (1997)) are such local performance measures which describe the behavior of a single step. The schema theorem also is a result which guarantees a certain behavior for one step. For Markoff processes (like evolutionary algorithms) the transition probabilities for one step determine the global behavior. However, the local performance measures are so-called "insufficient statistics" implying that in general it is not possible to deduce statements on the global behavior. Examples where these local performance measures give "wrong hints" are contained in Jansen and Wegener (2000b). Our global performance measures describe the behavior within reasonable time bounds. We think that the limit behavior is of much less interest. Even for state spaces like $S = \mathbb{R}^n$ we look for a good behavior within reasonable time bounds and not in the limit. Finally, interesting results have been obtained by modelling evolutionary algorithms as dynamical systems. However, this model implicitly works with infinite populations and one has to carefully investigate the difference between infinite populations, finite populations, and populations of reasonable size. Rabani, Rabinovich, and Sinclair (1998) have obtained first results how to control the difference between these cases.

4. SELECTED FUNCTIONS

Here we give an overview on the functions investigated in this paper as examples.

Definition 2 *A pseudo-boolean function $f : \{0,1\}^n \to \mathbb{R}$ is a degree-k function with N non-vanishing terms if it can be represented as*

$$f(x_1, \ldots, x_n) = \sum_{1 \le i \le N} w_i \prod_{j \in S_i} x_j$$

where $w_i \in \mathbb{R} - \{0\}$ and the size of the sets $S_i \subseteq \{1, \ldots, n\}$ is bounded above by k. Degree-1 functions are called linear and degree-2 functions are called quadratic.

The following two linear functions are of particular interest. They are the extreme examples of equal and strongly different weights.

Definition 3

$$\begin{aligned}&\mathit{ONEMAX}(x_1,\ldots,x_n) = ||x|| := x_1 + \cdots + x_n.\\ &BV(x_1,\ldots,x_n) = 2^{n-1}x_1 + 2^{n-2}x_2 + \cdots + 2x_{n-1} + x_n,\\ &\textit{where } BV \textit{ stands for binary value.}\end{aligned}$$

Unimodal functions are those functions where the global optimum is unique and can be reached from each point by 1-bit mutations.

Definition 4 *A pseudo-boolean function is called unimodal if it has a unique global optimum and all other search points have a Hamming neighbor with a larger fitness.*

We consider the special unimodal function LO (leading ones) and the class of path functions.

Definition 5
$LO(x_1,\ldots,x_n) = \max\{i | x_1 = \cdots = x_i = 1$ *and* $(i = n$ *or* $x_{i+1} = 0)\}$ *measures the length of the longest prefix consisting of ones only.*

Definition 6 *A path p starting at $a \in \{0,1\}^n$ is defined by a sequence of points $p = (p_0,\ldots,p_l)$ where $p_0 = a$ and $H(p_i,p_{i+1}) = 1$. A function $f : \{0,1\}^n \to \mathbb{R}$ is a path function with respect to the path p if $f(p_{i+1}) > f(p_i)$ for $0 \le i \le l-1$ and $f(b) < f(a)$ for all b outside the path.*

Definition 7 *SP (short path) is defined with respect to the path $p = (p_0,\ldots,p_n)$ where $p_i = 1^i 0^{n-i}$ by*

$$SP(x_1,\ldots,x_n) = \begin{cases} n+i & \textit{if } x = p_i \\ n - ||x|| & \textit{otherwise.}\end{cases}$$

Long path functions are defined on exponentially long paths. They were first introduced by Horn, Goldberg, and Deb (1994). Their long path functions admit the possibility of shortcuts namely a mutation flipping $O(1)$ bits and replacing p_i with p_j where $j-i$ is exponentially large. This can make them easy for evolutionary algorithms as shown by Rudolph (1997) who also introduced long path functions with the additional property that for each p_i there is at most one successor on the path with Hamming distance d if $d \le n^{1/2}$. Moreover, he has specified the fitness values outside the path to obtain a unimodal function which is difficult for the $(1+1)$EA. The exact definition is not necessary here. We call this function LP (long path).

Another interesting issue is the investigation of evolutionary algorithms in the presence of plateaus, i.e., connected subsets of the input space with constant fitness and with only few neighbored points with a better fitness. The following function is a good example for such a function.

Definition 8 *The function SPP (short path as plateau) is defined with respect to the path $p = (p_0, \ldots, p_n)$ where $p_i = 1^i 0^{n-i}$ by*

$$SPP(x_1, \ldots, x_n) = \begin{cases} 2n & \text{if } x = 1^n \\ n & \text{if } x = p_i, i < n \\ n - ||x|| & \text{otherwise.} \end{cases}$$

Finally, we introduce two special functions where the first one has the property of giving wrong hints and the second one is a special example for the investigation of the power of crossover.

Definition 9

$$\begin{aligned} TRAP(x_1, \ldots, x_n) &= \begin{cases} n & \text{if } x = 1^n \\ n - ||x|| & \text{otherwise.} \end{cases} \\ JUMP^m(x_1, \ldots, x_n) &= \begin{cases} ||x|| & \text{if } ||x|| < n - m \text{ or } ||x|| = n \\ n - ||x|| & \text{otherwise.} \end{cases} \end{aligned}$$

5. TAIL INEQUALITIES

Tail inequalities are useful to turn expected run times into upper time bounds which hold with overwhelming probability. Moreover, for many intermediate results, expected values sometimes are useless. A mutation flips on the average one bit, but we are interested in the probability of flipping more than k bits simultaneously. A random search point contains on the average $n/2$ ones, but what do we know about the probability of less than, e.g., $n/3$ ones? The simplest tail inequality is Markoff's inequality which is the basis of other tail inequalities. We present the result with its proof to show the underlying ideas.

Theorem 1 *(Markoff's inequality) Let X be a random variable taking only non-negative values. Then for all $t > 0$*

$$\mathrm{Pr}(\mathrm{X} \geq \mathrm{t}) \leq \mathrm{E}(\mathrm{X})/\mathrm{t}.$$

The result follows by estimating all X-values from $[0, t)$ by 0 and all other X-values by t:

$$\begin{aligned} E(X) &= \sum_x x \cdot \Pr(X = x) = \sum_{x<t} x \cdot \Pr(X = x) + \sum_{x \geq t} x \cdot \Pr(X = x) \\ &\geq 0 + t \cdot \Pr(X \geq t). \end{aligned}$$

In particular, $\Pr(X \geq 2E(X)) \leq 1/2$ or $\Pr(X < 2E(X)) \geq 1/2$ proving that with a probability of at least $1-(1/2)^k$ one out of k independent trials (or runs) has a value of less than $2E(X)$.

Other tail inequalities like Tschebyscheff's inequality and Chernoff's inequality are applications of Markoff's inequality. In the first case the probability of large deviations from the expected value is bounded, i.e., $\Pr(|X-E(X)| \geq \varepsilon)$. Before applying Markoff's inequality the inequality is replaced with the equivalent inequality $|X-E(X)|^2 \geq \varepsilon^2$. E.g., for $\varepsilon = 1/2$, the estimation of x^2 by 0 for $x < 1/4$ is more precise than the estimation of x by 0. For $x \geq 1/4$, x^2 is estimated by $1/16$ instead of estimating x by $1/4$. For small $x \geq 1/4$ this is a better estimate, but for larger x it gets worse. Hence, one may hope to get sometimes better results. Chernoff has used the "even more convex" function e^{tx} and $X \leq t$ is replaced with $e^{-sX} \geq e^{-st}$ for an appropriate value of s. We summarize the results (for the proofs see Motwani and Raghavan (1995)).

Theorem 2 *(Tschebyscheff's inequality) Let X be a random variable whose variance $V(X)$ exists. Then for all $\varepsilon > 0$*

$$\Pr(|\mathrm{X}-\mathrm{E(X)}| \geq \varepsilon) \leq \mathrm{V(X)}/\varepsilon^2.$$

(Chernoff's inequality) Let $X_1, \ldots, X_n$ be independent random variables taking values in $\{0,1\}$ and let $p_i = Pr(X_i = 1)$. Then $E(X) = p_1 + \cdots + p_n$ for $X = X_1 + \cdots + X_n$ and for $0 \leq \delta \leq 1$

$$\Pr(\mathrm{X} \leq (1-\delta)\mathrm{E(X)}) \leq \mathrm{e}^{-\mathrm{E(X)}\delta^2/2}$$

and for all $\delta > 0$

$$\Pr(\mathrm{X} > (1+\delta)\mathrm{E(X)}) \leq \left[\frac{\mathrm{e}^{\delta}}{(1+\delta)^{1+\delta}}\right]^{\mathrm{E(X)}}.$$

If $p_1 = \cdots = p_n = p$, X is binomially distributed and $E(X) = np$. The probability bounds are exponentially small with respect to n if δ is constant. The probability that a random string has less than $n/3$ ones can be bounded by $e^{-n/36}$ (choose $\delta = \frac{1}{3}$ implying that $(1-\delta)E(X) = n/3$). The probability that one chosen bit flips in n^2 steps less than $n/2$ times can be bounded by $e^{-n/8}$ ($E(X) = n, \delta = 1/2$). Even the probability that it flips less than $n - n^{3/4}$ times can be bounded by the asymptotically vanishing value of $e^{-n^{1/2}/2}$ ($E(X) = n, \delta = n^{-1/4}$). Hence, Chernoff's inequality gives concrete bounds (not only asymptotic ones) and it shows that the value of a binomially distributed random variable is with overwhelming probability very close to its expected value. This implies the following procedure. Produce an estimate by working in situations where Chernoff's inequality is applicable with expected values as "true" values. This leads to conjectures which often (with some weakening) can be proved rigorously.

6. THE COUPON COLLECTOR'S THEOREM

ONEMAX is perhaps the simplest function essentially depending on all variables and having a unique global optimum. Already Mühlenbein(1992) has proved an $O(n \log n)$ bound on the expected run time of the (1+1)EA on ONEMAX. This bound follows directly from a general upper bound technique presented in Section 8. What about a lower bound?

First, we present the coupon collector's theorem and then we argue why this result leads to an $\Omega(n \log n)$ lower bound on the expected run time of the (1+1)EA on ONEMAX. Consider the following experiment. There are n bins. Somebody throws randomly and independently balls into the bins and stops when each bin contains at least one ball. We are interested in the random number of balls in all bins together. The problem is equivalent to the problem of collecting coupons (pictures with, e. g., football players). There are n different types of coupons. We buy coupons until we have a complete collection of coupons. The expected time until we obtain a specific coupon equals n. However, we have to wait for the last coupon.

Theorem 3 *(Coupon collector's theorem)*

Let X be the random waiting time in the coupon collector's scenario. Then

$$\lim_{n\to\infty} Pr(X \le n \ln n - cn) = e^{-e^{c}}$$

and

$$\lim_{n\to\infty} Pr(X \ge n \ln n + cn) = 1 - e^{-e^{-c}}.$$

The proof of this result is contained in Motwani and Raghavan (1995). The theorem is described as limit behavior. The bounds for not two small n are close to these bounds in the limit. Such a result is also called a sharp threshold result, since the constant factor for the leading term $n \ln n$ is given exactly and the probability even of small deviations is very small.

Now we apply this result to the analysis of the (1+1)EA on ONEMAX (Droste, Jansen, and Wegener (1998a)). The probability that the first search point has less than $2n/3$ ones is overwhelmingly large (Chernoff's bound). It is necessary that each 0-bit flips at least once. We investigate

$n \ln n + cn$ steps and only $n/3$ bit positions which are initialized by 0. Again we can apply Chernoff's bound. With large probability, the number of flipping bits is bounded above by $\frac{1}{3} n \ln n + cn$. We consider the $n/3$ bit positions as bins and the flipping bits as balls. However, the balls are not totally independent. If during one step at least two bits flip, these bits are at different positions. It is possible to bound the difference between the real situation and the situation of the coupon collector's theorem. This also is a general rule. One often can apply results from probability theory — but not in their pure form.

Theorem 4 *The expected run time of the (1+1)EA on ONEMAX is bounded above by* $e \cdot n \cdot (\ln n + 1)$. *For each constant* $d > 0$ *there is a constant* c *such that the success probability of the (1+1)EA on a function with a unique global optimum within* $n \ln n - cn$ *steps is bounded above by* d.

With a more careful analysis one will even get better lower bounds for ONEMAX. However, the crucial fact is that the coupon collector's theorem leads to a lower bound for a large class of functions.

7. THE GAMBLER'S RUIN PROBLEM

The classical gambler's ruin problem is the following one. Alice owns a dollars and Bob owns b dollars. They are throwing coins and Alice pays Bob one dollar if the coin shows head and receives one dollar from Bob for tails. The game stops if somebody gets ruined. The problem is to compute the probability that Alice gets ruined and how long it does take until the game stops.

We may use ideas from the solution of the gambler's ruin problem in order to analyze the $(1+1)$EA on SPP, the short path with $n+1$ points where n of them have the same fitness (they belong to the plateau) and the last point is the global optimum. Since the function is defined as $n-$ONEMAX outside the path, it follows from the analysis of ONEMAX that the $(1+1)$EA needs an expected time of $O(n \log n)$ to reach the plateau. With overwhelming probability, the plateau is reached close to 0^n. Hence, we have to analyze the behavior of the $(1+1)$EA on the plateau. Only other points from the plateau and the global optimum are accepted. The situation is similar to the gambler's ruin problem — with some differences. The probability of accepting a different search point equals $\Theta(\frac{1}{n})$ (otherwise, no bit is flipped or a search point outside the path is reached). The conditional probability of an accepted step of length 1 is $1 - \Theta(\frac{1}{n})$, and, for $j \geq 2$, the conditional probability of a step length of j equals $\Theta((\frac{1}{n})^{j-1})$. One has to control with some technical effort the influence of steps of length $j \geq 2$. The steps of length 1 from $1^i 0^{n-i}$, $0 < i < n$, reach the points $1^{i-1}0^{n-i+1}$ and $1^{i+1}0^{n-i-1}$ each with probability 1/2 – like in the gambler's ruin problem. Nobody is ruined at 0^n, but all accepted new strings have more ones.

The main part of the analysis is easy. In order to generalize the result, we consider a plateau length l and a phase of length cnl^2, c a large enough constant. By Chernoff's bound, the number of accepted steps is with overwhelming probability at least $c'l^2$ (where c' can be chosen as any constant by increasing c). Here we ignore all accepted steps whose length is at least 2 (for these details see Jansen and Wegener (2000b)). We want to estimate the probability that the number of "increasing steps", i.e., steps of length 1 which increase the number of ones, is at least $c'l^2/2 + l/2$. Such an event implies that we have reached the global optimum. By symmetry, the probability of at most $c'l^2/2$ increasing steps, equals 1/2. The binomial distribution with parameters m and 1/2 has its highest value at $\lfloor m/2 \rfloor$ where the probability is bounded above by $\alpha m^{-1/2}$ for some constant α. Hence, the probability of at least $c'l/2$ and at most $c'l^2/2 + l/2$ increasing steps is bounded above by $(l/2 + 1) \cdot \alpha \cdot (c'l^2/2)^{-1/2}$ which can be bounded by 1/4 for large enough c'. This implies a success probability of at least 1/4 and the expected number of phases before a first success is at most 4. Hence, the expected time to pass a plateau which is a path of length l is bounded above by $O(nl^2)$. If the fitness is increasing along the path, the expected run time equals $\Theta(nl)$ (see Section 8). Obviously, it is easier to follow a path with increasing fitness, but even a path giving no hints is no real obstacle for the (1+1)EA. The following theorem describes the result for SPP where $l = n$.

Theorem 5 *The expected run time of the* $(1+1)$*EA on SPP is bounded above by* $O(n^3)$*. The success probability for* n^4 *steps is bounded below by* $1 - 2^{-\Omega(n)}$.

8. UPPER BOUNDS BY ARTIFICIAL FITNESS LEVELS

The following upper bound technique is based on a suitable partition of the search space according to the fitness function.

Definition 10 *For* $A, B \subseteq \{0,1\}^n$ *and* $f : \{0,1\}^n \to \mathbb{R}$ *the relation* $A <_f B$ *holds if* $f(a) < f(b)$ *for all* $a \in A$ *and* $b \in B$*.* $A <_f$*-partition of* $\{0,1\}^n$ *is a partition of* $\{0,1\}^n$ *into non-empty sets* $A_1, \ldots, A_m$ *such that* $A_1 <_f A_2 <_f \cdots <_f A_m$ *and all* $a \in A_m$ *are global optima.*

Lemma 1 *Let $A_1, \ldots, A_m$ be a $<_f$-partition, let $p(A_i)$ be the probability that a randomly chosen search point belongs to A_i, let $s(a)$ be the probability that a mutation of $a \in A_i$ belongs to $A_{i+1} \cup \cdots \cup A_m$, and let $s_i = \min\{s(a) | a \in A_i\}$. Then*

$$E(X_f) \leq \sum_{1 \leq i \leq m-1} p(A_i)(s_i^{-1} + \cdots + s_{m-1}^{-1}) \leq s_1^{-1} + \cdots + s_{m-1}^{-1}.$$

The proof of Lemma 1 is very simple. The second inequality is trivial and the first inequality is based on the law of total probability. When starting in A_i, the expected time to leave A_i is bounded above by s_i^{-1} and we have to leave only some of the sets $A_i, \ldots, A_{m-1}$. Nevertheless, this result is surprisingly powerful. However, it is crucial to choose an appropriate $<_f$-partition. We start with simple applications leading to asymptotically optimal bounds.

For ONEMAX let A_i contain all points with i ones, $0 \leq i \leq n$. For $x \in A_i$ there are $n - i$ 1-bit mutations leading to A_{i+1} (if $i < n$). Hence, $s_i \geq (n-i)\frac{1}{n}(1 - \frac{1}{n})^{n-1} \geq e^{-1} \cdot (n-i) \cdot \frac{1}{n}$ and

$$s_0^{-1} + \cdots + s_{n-1}^{-1} \leq e \cdot n \left(\frac{1}{1} + \frac{1}{2} + \cdots + \frac{1}{n} \right) \leq e \cdot n \cdot (\ln n + 1).$$

This bound can be improved by considering the probabilities $p(A_i)$, but the improvement saves only a linear additive term. A corresponding lower bound has been presented in Section 6.

For LO let A_i contain all points with fitness i, i.e., starting with i but not with $i + 1$ ones. For $x \in A_i$ there is one 1-bit mutation leading to A_{i+1} (if $i < n$). Hence, $s_i \geq e^{-1} \cdot \frac{1}{n}$ and we obtain the upper bound $e \cdot n^2$ where l is the length of the path. A corresponding lower bound will be presented in Section 9.

For path functions we assume that the values outside the path are chosen in such a way that the path is reached quickly (e.g., $n-$ONEMAX if 0^n is the source of the path). The expected run time for the first point on the path is denoted by $t(n)$. For the rest of the search let A_i contain the ith point of the path. Since there is by definition a 1-bit mutation to the successor on the path, we get an upper bound for the expected run time of size $e \cdot n \cdot l + t(n)$. A corresponding lower bound will be presented in Section 9.

These considerations can be generalized to all unimodal functions, since there is always a 1-bit mutation improving the fitness. If the unimodal function f takes $w(f)$ different values, the expected run time of the algorithm can be bounded above by $e \cdot n \cdot (w(f) - 1)$. The function BV (binary value) is unimodal and takes the maximal number of 2^n different values. The upper bound $e \cdot n \cdot (2^n - 1)$ on the expected run time is correct. However, this bound seems to be far from optimal. Sitting in 01^{n-1}, it is very likely that the next accepted step improves the fitness by much more than 1.

We investigate the class of degree-k functions f with N non-vanishing weights which are all positive (Wegener and Witt (2000)). We number the N positive weights in non-increasing order: $w_1 \geq w_2 \geq \cdots \geq w_N > 0$. Then we use the following $<_f$-partition $A_0, \ldots, A_N$ where

$$A_i = \{x | w_1 + \cdots + w_i \leq f(x) < w_1 + \cdots + w_{i+1}\}$$

for $0 \leq i \leq N-1$ and

$$A_N = \{x | f(x) = w_1 + \cdots + w_N\}.$$

For $x \in A_i$ there is some $j \in \{1, \ldots, i+1\}$ such that the w_j-term is not activated, i.e., not all bits belonging to S_j are equal to 1. The mutation where all 0-bits belonging to S_j flip activates the w_j-term. This increases the f-value by at least w_j. Here it is essential that all weights are non-negative. By definition of A_i, the resulting search point belongs to $A_{i+1} \cup \cdots \cup A_N$. Since we consider a mutation of at most $|S_j| \leq k$ bits, $s_i^{-1} \leq e \cdot n^k$ leading to an upper bound on the expected run time of $e \cdot n^k \cdot N$. For linear functions, negative weights can be replaced with positive weights without changing the behavior of the (1+1)EA. Variables x_i with $w_i < 0$ are replaced with $\overline{x}_i = 1 - x_i$. For degree-k functions, $k \geq 2$, such a replacement can create new negative weights. Since $N \leq n$ for linear functions, we obtain an $e \cdot n^2$ upper bound for all linear functions and the upper bound for BV is decreased from exponential to quadratic. An even better upper bound will be presented in Section 10. We summarize our results.

Theorem 6 *The following upper bounds hold for the expected run times of the (1+1)EA:*

- $e \cdot n \cdot (\ln n + 1)$ *for ONEMAX,*
- $e \cdot n^2$ *for all linear functions,*
- $e \cdot n^k \cdot N$ *for all degree-k functions with N non-vanishing weights which all are positive,*
- $e \cdot n^2$ *for LO,*
- $e \cdot n \cdot l$ *for path functions on paths of length l if the search starts on the path,*
- $e \cdot n \cdot N$ *for all unimodal functions taking at most $N + 1$ different values.*

9. LOWER BOUNDS BY ARTIFICIAL FITNESS LEVELS

We look for a lower bound result corresponding to the upper bound of Lemma 1. We use the notations of Lemma 1. If the initial search point belongs to A_i, $i < m$, the search has to leave A_i. Let $\mu_i = \max\{s(a) | a \in A_i\}$. Then the expected time to leave A_i is bounded below by μ_i^{-1} leading to the following result.

Lemma 2 $E(X_f) \geq \sum_{1 \leq i \leq m-1} p(A_i)\mu_i^{-1}$.

Lemma 2 is less powerful than Lemma 1. The reason is that we usually do not reach a global optimum from A_i. Hence, we should investigate how many A-levels we usually pass in one step. Such an approach has been realized for LO and path functions by Droste, Jansen, and Wegener (1998b).

For LO we investigate the stochastic process behind the (1 + 1)EA more carefully. If we have produced a string x where $\mathrm{LO}(x) = i$, then $x_1 = \cdots = x_i = 1$, $x_{i+1} = 0$, and $(x_{i+2}, \ldots, x_n)$ is a random string. The last property is easy to prove and essential. If a step increases the fitness, we know that none of the first i bits is flipping, the $(i+1)$st bit flips and the new suffix $(x'_{i+2}, \ldots, x'_n)$ again is a random string. We have k "free-riders" if exactly k leading bits of $(x'_{i+2}, \ldots, x'_n)$ are ones. The production of free-riders can be described by the following stochastic process. We have a random bit string $a = (a_1, a_2, \ldots)$. The number of ones between two consecutive zeros describes the number of free-riders. The probability that we have more than $2n/3$ free-riders during $n/3$ fitness-increasing steps (including the initialization) is equal to the probability that a random string from $\{0, 1\}^n$ contains more than $2n/3$ ones. By Chernoff's bound this probability is exponentially small. Hence, we have to wait with overwhelming probability for at least $n/3$ fitness-increasing steps. The probability of a fitness-increasing step is bounded above by $1/n$, since one special bit has to flip. Another application of Chernoff's bound shows that with overwhelming probability $n^2/6$ steps are not enough to have $n/3$ fitness-increasing steps.

In the following we investigate the long path function LP. It is sufficient to know that the path length is $l = \Theta(n^{1/2}2^{n^{1/2}})$. Each point p_i on the path has for each $d \leq n^{1/2}$ at most one successor p' on the path such that $H(p_i, p') = d$. If $i + d \leq l$, $H(p_i, p_{i+d}) = d$. Moreover, the fitness outside the path is defined in such a way that we can assume that the path is reached for the first time in its first half. The idea is to estimate the expected progress along the path during one step.

The probability that at least $n^{1/2}$ bits flip simultaneously, can be bounded by $2^{-\Omega(n^{1/2}\log n)}$. For such cases we estimate the progress by the simple upper bound l leading to a contribution of $2^{-\Omega(n^{1/2}\log n)}$ to the expected progress. If less than $n^{1/2}$ bits flip simultaneously, only one special k-bit mutation is accepted and leads to a progress of k on the path. The probability of a special k-bit mutation is bounded above by n^{-k}. Hence, the expected progress in one step is bounded above by

$$\sum_{1\le k<n^{1/2}} k\cdot n^{-k} + 2^{-\Omega(n^{1/2}\log n)} \le \frac{2}{n} + 2^{-\Omega(n^{1/2}\log n)}.$$

The expected progress within $l\cdot n/5$ steps is bounded above by $(\frac{2}{5}+o(1))l$ and we need a progress of $l/2$. Markoff's inequality proves a bound of $\Omega(l\cdot n)$ on the expected run time of the $(1+1)$EA on LP.

Theorem 7 *The following lower bounds hold on the expected run times of the* $(1+1)$*EA:*

– $n^2/6 - o(n^2)$ *for LO.*

– $\Omega(ln)$ *for long path functions where* $l = O(n^{1/2}2^{n^{1/2}})$ *not allowing short cuts by at most* $n^{1/2}$ *flipping bits and the property that the path is reached with constant positive probability in the first half.*

Such a long path function where $l = O(n^{1/2}2^{n^{1/2}})$ has been defined proving that unimodal functions can be hard for the $(1+1)$EA.

10. POTENTIAL FUNCTIONS

The general upper bound for unimodal functions leads to a bad upper bound for BV (binary value). The design of a problem-specific $<_f$-partition allows a simple proof of a quadratic upper bound. The same holds for all linear functions. We have a gap between the $\Omega(n\log n)$ lower bound from Section 6 and the $O(n^2)$ upper bound from Section 8. Droste, Jansen and Wegener (1998a) have improved the upper bound to $O(n\log n)$. It is difficult to control the Hamming distance to the global optimum which, under the assumption $w_1\ge\cdots\ge w_n>0$, equals 1^n. Hence, the idea is to measure the progress with respect to some well-chosen measure. The idea of $<_f$-partitions is already a step into this direction. These partitions have the advantage that only strings from $A_i\cup\cdots\cup A_m$ can be accepted from $x\in A_i$. Hence, the function $g(t)$ describing the index of the A-set containing the actual search point after t steps is a non-decreasing function. The asymptotically exact bound for linear functions has been obtained only by using a so-called potential function where it is quite likely that the value of the potential decreases in some steps. In particular, these potential functions do not depend on the special values of the linear function f.

The new method is easier to explain for BV. The potential function equals ONEMAX, the number of ones in the string. To be precise, the following is important:

– the (1+1)EA uses the fitness function f in order to decide whether the old search point is replaced with its mutant,

– the people analyzing the (1+1)EA use the potential function p to measure the progress.

A step is called successful if the mutant x' is different from its parent x and replaces x. The crucial step is to estimate the expected number of steps until the (1+1)EA produces from x with $p(x) = i$ a search point x' with a higher p-value. We distinguish between successful and unsuccessful steps. Knowing the expected number of successful steps it is in this special situation not too difficult to bound the expected number of unsuccessful steps.

BV has the advantage that a successful step has a simple description. A step is successful iff the leftmost flipping bit is a 0. All other bits flip with mutation probability $1/n$ even under the assumption that the step is successful. Knowing that x contains i ones we do not know how many are to the right of the leftmost flipping bit which is a zero. Hence, we pessimistically analyze a provable slower Markoff process assuming that only the leftmost flipping bit is a flipping 0 and that $n - i$ 1-bit positions have the chance to flip to 0. Now we run into difficulties. The expected progress with respect to the potential function can only be bounded below by $1 - \frac{n-i}{n} = \frac{i}{n}$ leading to an expected number of at most n/i successful steps before the p-value has increased. The last conclusion follows from Wald's identity. For BV, a simple trick works. We restrict our attention to the first $n/2$ partitions. The behavior of the second half of the n bits has no influence on the decision whether the actual first half of the string is changed by the (1+1)EA. The expected progress is now bounded below by $1/2$ leading to an expected number of at most two successful steps before the p-value has increased. This leads after some calculations on the number of unsuccessful steps to an $O(n \log n)$ bound for the time until the actual search point starts with $n/2$ ones. The second half can be treated in the same way. Only the number of unsuccessful steps has to be multiplied by $(1 - \frac{1}{n})^{n/2} \approx e^{-1/2}$, the probability that no bit of the first half flips. This describes the proof method for the special case of BV. There are many places where the very special properties of BV have been used.

Surprisingly, there is one special linear function serving as a potential function for all linear functions. This function is

$$p(x) = 2 \cdot (x_1 + \cdots + x_{n/2}) + 1 \cdot (x_{n/2+1} + \cdots + x_n).$$

This potential function is somehow a compromise between equal weights (ONEMAX) and very different weights (BV). By an involved and tedious case inspection, it can be shown that the expected number of successful steps until the value of the potential function p has increased is bounded above by a constant (independent from the starting point). Then the methods discussed for the analysis of the (1+1)EA on BV can be generalized to lead to the proposed bound.

Theorem 8 *The expected run time of the (1+1)EA on an arbitrary linear function is bounded above by* $O(n \log n)$*.*

11. INVESTIGATIONS OF TYPICAL RUNS

The sections on tail inequalities and the coupon collector's theorem have shown that stochastic experiments or stochastic processes have a "typical global behavior", although the local behavior is unpredictable. Tossing coins one has no idea about the outcome of a toss. Tossing independently many coins we also have no idea about a certain coin, but we have very tight bounds describing the number of heads — if we allow an exponentially small error probability. The same is true for the coupon collector. Obtaining the next coupon she or he can only hope for a coupon which she or he does not hold yet. Obtaining $g(n)$ coupons, for each value of $g(n)$ outside a small interval one can be almost sure to obtain all different coupon types or one can be almost sure that some coupons are missing. The same holds for evolutionary algorithms. The behavior within one step or a few steps has a large uncertainty, although one can obtain bounds for the global behavior which have a very small error probability.

The idea is to investigate search phases of well-chosen lengths. During each phase we expect a certain behavior of the (1+1)EA. Runs of the (1+1)EA fulfilling the aims of all phases are called "typical". We should choose our definition in a way that the run time of a typical run has properties we expect to happen with overwhelming probability. The essential point is to prove an upper bound on the failure probability, i.e., the probability of a non-typical run. This can be done by adding the failure probabilities for the different phases. Sometimes, we get better results if we ensure that the different failure events are independent. Otherwise, we can bound the failure probability for the ith phase using the assumption that the first $i - 1$ phases were free of failures.

A simple application of this method is the analysis of the (1+1)EA on TRAP. The simple upper bound n^n holds for all pseudo-boolean functions (Droste, Jansen, and Wegener (1998b)). We expect that this bound is quite tight for TRAP, since TRAP gives everywhere (except at the global optimum) wrong hints. However, the initial search point has approximately $n/2$ ones and the probability that mutation creates the global optimum is approximately $n^{-n/2}$. The expected waiting time for an event with such a success probability is approximately $n^{n/2}$ – only the square root of n^n.

Now we consider a typical run consisting of three phases:

- the initialization phase, we expect an initial string with less than $(2/3)$ ones,
- the phase of the first $cn^2 \log n$ steps, we expect to have 0^n as last actual search point,
- the phase of the remaining steps starting with 0^n, the expected run time equals n^n.

Without failures during the first two phases we have 0^n as actual point and accept only 1^n as new search point. The probability that mutation produces 1^n from 0^n equals n^{-n} leading to an expected waiting time of n^n. The failure probability of the first phase is $2^{-\Omega(n)}$ (Chernoff's bound). If there was no failure in the first phase, a failure in the second phase implies that we either flip at least $n/3$ steps in one step (failure type 1) or the optimization of $n -$ ONEMAX takes more than $cn^2 \log n$ steps (failure type 2). The probability of a failure of type 1 in one step is bounded above by $\frac{1}{(n/3)!} = 2^{-\Omega(n \log n)}$ leading to an upper bound of $cn^2(\log n)2^{-\Omega(n \log n)} = 2^{-\Omega(n \log n)}$ for the whole phase. We know that the expected run time of the (1+1)EA on $n -$ ONEMAX is bounded above by $c'n \log n$ for some constant c'. Let $c = 2c'$. Then we get n independent subphases of length $2c'n \log n$ each. The probability that a subphase is unsuccessful is bounded above by $1/2$ (Markoff's inequality) leading to a 2^{-n} bound for the probability of type 2 failures. (It is possible to obtain even better bounds.) Altogether, with probability $1 - 2^{-\Omega(n)}$ no failure occurs leading to an expected run time of n^n. This proves the following result.

Theorem 9 *The expected run time of the (1+1)EA on TRAP is bounded above by n^n and below by $(1 - 2^{-\Omega(n)})n^n$.*

A more sophisticated application of this method has been presented by Jansen and Wegener (1999). Crossover is known as one of the fundamental operators of evolutionary algorithms. Nevertheless, Jansen and

Wegener (1999) were the first to prove rigorously for a function, namely JUMP^m for $m = \lfloor \log n \rfloor$, that the expected run time of evolutionary algorithms without crossover grows superpolynomially ($n^{\Omega(\log n)}$) while a steady-state genetic algorithm with population size n and the small crossover probability $p_c = 1/(n \log n)$ has a polynomial expected run time.

Theorem 10 *Evolutionary algorithms without crossover need for $JUMP^m$, $m = \lfloor \log n \rfloor$, with large probability superpolynomial time. A steady-state genetic algorithm with population size n and probability $p_c = 1/(n \log n)$ for uniform crossover has an expected run time on $JUMP^m$ which is bounded above by $O(n^3 \log n)$.*

We discuss the problems with the proof of an upper bound for the steady-state EA. It seems easy to obtain a population consisting only of individuals with exactly m zeros (or even an optimal individual). Uniform crossover on two individuals which have m zeros each and the zeros at different positions has a probability of exactly $2^{-2m} \geq 1/n^2$ to produce the global optimum 1^n and the probability that mutation does not destroy 1^n is approximately e^{-1}. The expected waiting time increases to $O(n^3 \log n)$ because of the small crossover probability. This small probability simplifies the control of the hitchhiking effect. If we choose two random strings with m zeros, it is very likely to have the zeros at different positions. Crossover and selection destroy the independency of the individuals of the initial population. It gets more likely that individuals share zeros. The proof of Theorem 10 is possible with the following definition of a typical run. Since the estimation of the failure probabilities is too complicated to be presented here in detail, we also do not present the chosen phase lengths.

We expect that after the first phase either we have found the optimum or that all individuals have exactly m zeros. The failure probability can be estimated using a generalization of the coupon collector's theorem and Chernoff's bound. After the second phase we expect that either we have found the optimum or all individuals still have exactly m zeros and for each bit position the number of individuals with a zero at this position is bounded above by $n/(4m)$. We sum the exponentially small failure probabilities for the n different bit positions. The small crossover probability makes it possible to consider crossover as a bad event which increases the number of individuals with a zero at the considered position. For the third phase we expect that for no bit position and no point of time the number of individuals with a zero gets larger than $n/(2m)$. This implies a probability of at least $1/2$ that two individuals have no zero in common and crossover can work. Hence, we expect to find the optimum during the third phase.

Conclusion

The analysis of evolutionary algorithms can be based on methods from the analysis of classical deterministic and randomized algorithms. Some tools which seem to be of large value for evolutionary algorithms have been presented, discussed, and applied. This has led to a number of results on the expected run time and the success probability of evolutionary algorithms.

References

Bäck, T. (1993). Optimal mutation rates in genetic search. Proc. of 5th ICGA (Int. Conf. on Genetic Algorithms), 2–8.

Bäck, T. (1998). An overview of parameter control methods by self-adaptation in evolutionary algorithms. Fundamenta Informaticae 35, 51–66.

Bäck, T., Fogel, D. B., and Michalewicz, Z. (Eds). (1997). *Handbook of Evolutionary Computation.* Oxford Univ. Press.

Beyer, H.-G. (1996). Toward a theory of evolution strategies: Selfadaptation. Evolutionary Computation 3, 311–347.

Cormen, T. H., Leiserson, C. E., and Rivest, R. L. (1990). *Introduction to Algorithms.* MIT Press.

Droste, S., Jansen, T., and Wegener, I. (1998a). A rigorous complexity analysis of the (1+1) evolutionary algorithm for separable functions with Boolean inputs. Evolutionary Computation 6, 185–196.

Droste, S., Jansen, T., and Wegener, I. (1998b). On the optimization of unimodal functions with the (1+1) evolutionary algorithm. Proc. of PPSN V (Parallel Problem Solving from Nature), LNCS 1498, 13–22.

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

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

Hochbaum, D. S. (1997). *Approximation Algorithms for NP-Hard Problems.* PWS Publ. Co., Boston.

Holland, J. H. (1975). *Adaptation in Natural and Artificial Systems.* The University of Michigan Press.

Horn, J., Goldberg, D. E., and Deb, K. (1994). Long path problems. Proc. of PPSN III (Parallel Problem Solving from Nature), LNCS 866, 149–158.

Jansen, T., and Wegener, I. (1999). On the analysis of evolutionary algorithms – a proof that crossover really can help. Proc. of ESA '99 (European Symp. on Algorithms), LNCS 1643, 184–193.

Jansen, T., and Wegener, I. (2000a). On the choice of the mutation probability for the (1+1)EA. Proc. of PPSN VI (Parallel Problem Solving from Nature), LNCS 1917, 89–98.

Jansen, T., and Wegener, I. (2000b). Evolutionary algorithms — how to cope with plateaus of constant fitness and when to reject strings of the same fitness. Submitted to IEEE Trans. on Evolutionary Computation.

Motwani, R., and Raghavan, P. (1995). *Randomized Algorithms.* Cambridge Univ. Press.

Mühlenbein, H. (1992). How genetic algorithms really work. I. Mutation and hillclimbing. Proc. of PPSN II (Parallel Problem Solving from Nature), 15–25.

Rabani, Y., Rabinovich, Y., and Sinclair, A. (1998). A computational view of population genetics. Random Structures and Algorithms 12, 314–334.

Rudolph, G. (1997). How mutations and selection solve long path problems in polynomial expected time. Evolutionary Computation 4, 195–205.

Schwefel, H.-P. (1995). *Evolution and Optimum Seeking.* Wiley.

Wegener, I., and Witt, C. (2000). On the behavior of the (1+1) evolutionary algorithm on quadratic pseudo-boolean functions. Submitted to Evolutionary Computation.