A RIGOROUS COMPLEXITY ANALYSIS OF THE (1 + 1)-EVOLUTION STRATEGY FOR SEPARABLE FUNCTIONS WITH BOOLEAN INPUTS

Stefan Droste, Thomas Jansen, and Ingo Wegener

FB Informatik, LS 2, Univ. Dortmund, 44221 Dortmund, Germany

droste, jansen, wegener@ls2.cs.uni-dortmund.de

Abstract

Evolutionary algorithms (EAs) are heuristic randomized algorithms which, by many impressive experiments, have been proven to behave quite well for optimization problems of various kinds. In order to improve our abilities in applying these algorithms three approaches should be followed in parallel. First, experiments for benchmark and practical problems have to be performed. Second, explanations about the behavior of EAs can be obtained by an analysis based on reasonable assumptions. Third, also a rigorous analysis without any unproven assumptions is necessary to establish in future a theory of EAs. Here, for the first time, a rigorous complexity analysis of the $(1 + 1)$ evolutionary algorithm for separable functions with Boolean inputs is given. Different mutation rates are compared and the use of the crossover operator is investigated. The main contribution is not the result that the expected run time of the $(1 + 1)$ evolutionary algorithm is $\Theta(n \ln n)$ for separable functions with n variables but the presentation of the methods how this result can be proven rigorously.

1 INTRODUCTION

Many experiments have shown that evolutionary algorithms (EAs) are a useful tool for the solution or approximate solution of optimization problems. Evolutionary algorithms are a general algorithmic concept that includes genetic algorithms (GAs), evolution strategies (ESs), and evolutionary programming (EP). For a description of the rich world of the different types of EAs we refer to the monographs of Fogel (1995), Goldberg (1989), Rechenberg (1994), and Schwefel (1995).

It is a general experience with algorithmic tools and strategies (from divide-and-conquer, branch-and-bound, dynamic programming, greedy algorithms to local search, simulated annealing, plane cutting or tabu search) that the development of these tools and strategies has to be guided by experiments and by theoretical analysis (e. g., see the monographs of Cormen, Leiserson, and Rivest (1990) and Sedgewick (1991) or the handbook of theoretical computer science (van Leeuwen (1990)). Such an analysis is quite simple for divide-and-conquer algorithms and becomes very involved for more heuristic approaches. Then also an analysis based on reasonable assumptions is of some interest. But those investigations may be misleading. Only a rigorous analysis based on no assumptions leads to proven results which can be the basis of a theory of EAs. In the present situation "theory" is far behind "experimental knowledge". To build up a theory problems have to be investigated which already are well understood from a practical point of view.

In this paper we present a contribution of the type described above. Separable functions with Boolean inputs (GAs usually work on Boolean strings) are investigated.

Definition 1: A function $f : \{0,1\}^n \to \mathbb{R}$ is called separable if it can be written in the form

$$f(x) = f(x_1, \ldots, x_n) = \sum_{1 \le i \le n} g_i(x_i).$$

For a separable function it is possible to optimize each variable individually. This can be done deterministically in linear time. But we analyze evolutionary algorithms which are not adopted to the class of separable functions. Then we may concentrate w. l. o. g. on the following situation.

- f has to be maximized (otherwise consider $-f$),
- $g_i(x_i) = w_i x_i$ for some $w_i \in \mathbb{R}$ (the general case is $g_i(x_i) = w_i x_i + \tau_i$, since $x_i \in \{0,1\}$, and the constant term has no influence on the optimization process),
- the weights w_i, $1 \le i \le n$, are positive (otherwise replace x_i by $1 - x_i$),
- the weights are integers (real numbers may be approximated by rational ones which can be multiplied by their smallest common denominator),
- the weights are sorted, i. e., $w_1 \ge \cdots \ge w_n$.

Our assumptions imply that the all-one input $(1, \ldots, 1)$ is the only solution of the optimization problem.

Mühlenbein and Schlierkamp-Voosen (1993a, 1993b) have argued that the crossover operator does not help in this situation. Therefore, we mainly analyze the following $(1+1)$ evolutionary algorithm (EA).

Algorithm 2:

1. The mutation rate $p(n) \in (0,1)$ is fixed.
2. Choose randomly an initial vector $x \in \{0,1\}^n$.
3. Repeat the following step: Compute x' by mutating independently each bit x_i with probability $p(n)$ and replace x by x' iff $f(x') \ge f(x)$.

We are interested in the expected number of steps until Algorithm 2 reaches the optimal vector $x^* = (1, \ldots, 1)$ for the first time. The following two functions are of particular interest.

Definition 3: i) The function $COUNT$ is the separable function where $w_1 = \cdots = w_n = 1$, i. e., $COUNT(x_1, \ldots, x_n) = x_1 + \cdots + x_n$.

ii) The function BIN is the separable function where $w_i = 2^{n-i}$, i. e., BIN interprets $(x_1, \ldots, x_n)$ as the binary representation of an integer.

These two functions are in the following sense the two extreme separable functions. The Hamming distance $H(x, x')$ of $x, x' \in \{0,1\}^n$ is the number of indices i where $x_i \neq x'_i$. A mutation step from x to x' is successful for $COUNT$ iff at least as many zeros flip to ones than vice versa. A successful mutation step does not increase the Hamming distance to the optimal vector x^*. A mutation step for BIN is successful iff the leftmost flipping bit flips from 0 to 1. E. g., $x' = (1, 0, \ldots, 0)$ can replace $x = (0, 1, \ldots, 1)$ although $H(x^*, x') = n - 1 > 1 = H(x^*, x)$. Hence, it should not be surprising that the analysis is much simpler for $COUNT$ than for BIN.

In Section 2 we argue why and how a heuristic analysis may lead to wrong results and we refer to known results. In Section 3 we analyze the $(1+1)$ evolutionary algorithm for BIN with respect to the recommended mutation rate $1/n$. In Section 4 we investigate other mutation rates and in Section 5 all other separable functions. Finally, we consider in Section 6 the use of larger populations and the crossover operator.

2 ON A HEURISTIC ANALYSIS OF EVOLUTIONARY ALGORITHMS

Salomon (1996) presents a lot of results on GAs, in particular for the case of a coordinate rotation of benchmark functions. His analysis of Algorithm 2 for separable functions is only a heuristic one. He argues as follows. "A mutation probability $p_m \leq 1/n$ implies that, on average, a GA modifies at most one parameter per offspring. By this means, such GAs decompose the optimization of a decomposable n-dimensional function into a sequence of n independent one-dimensional tasks. Since the complexity of the one-dimensional case is $O(1)$, convergence in the n-dimensional case is given by the probability p that all parameters x_i, $1 \leq i \leq n$, have been chosen at least once." Then it is a standard calculation to prove that this happens for $p_m = 1/n$, on average and with high probability, in time $\Theta(n \log n)$. (The notation Θ describes simultaneously a lower and an upper bound of the same growth rate.)

Let us discuss the case of the mutation probability $p_m = 1/n$. The random number of mutating bits (in one step) is described by a binomial distribution with parameters n (number of individuals) and $1/n$ (probability of "success"). This distribution converges for increasing n to the Poisson distribution with parameter $\lambda = n(1/n) = 1$ (for basic results from probability theory see Feller (1968)). The average number of mutating bits equals exactly 1. But it is wrong to conclude that we can consider, on average, the case where always one bit tries to mutate. This is the correct analysis of another Algorithm 2^* where in Step 3 always one bit is chosen according to the uniform distribution and mutated. With Algorithm 2^* it is impossible to escape from local maxima. The evolutionary algorithm often takes advantage from the fact that sometimes no bit and sometimes several bits flip. The Poisson distribution with parameter $\lambda = 1$ takes the value k (number of flipping bits) with probability $e^{-1}/k!$, i. e., probability $e^{-1} \approx 0.368$ for $k = 0$ and $k = 1$, $e^{-1}/2 \approx 0.184$ for $k = 2$, $e^{-1}/6 \approx 0.061$ for $k = 3$, and so on. In particular, the probability that more than one bit flips is not negligible.

We discuss the danger of non-rigorous reasoning. Nothing changes if no bit tries to flip. Hence, we have to consider only steps where at least one bit tries to flip. Under the assumption of one flipping bit the random number of *additional* bits which try to flip is described by a binomial distribution with parameters $n-1$ and $1/n$ or, asymptotically, by a Poisson distribution with parameter $\lambda = 1$. Hence, the expected number of additional flipping bits equals $(n-1)/n \approx 1$.

Considering $x = (0, 1, \ldots, 1)$ and the function $COUNT$, the first bit has to flip if something may change. But by the misleading argument that (on average) always another bit tries to flip we *never* reach the optimal vector x^*. Considering the function BIN for the same vector x the reasoning of Salomon (1996) implies that we only have to wait for the first time where the first bit tries to flip. The expected time of this event can easily be calculated as n. If we argue that, on average, one other bit tries to flip, after, on average, n steps we obtain a vector with one 0 which, on average, is situated at position $n/2$. Then we have to wait on average n steps until this bit tries to flip. With probability 1/2 the other flipping bit is left from this bit and nothing happens and with probability 1/2 we obtain another vector with one 0 which, on average, is at position $(3/4)n$. Following this reasoning the average time to reach the optimal vector is $\Theta(n^2)$.

We conclude that non-rigorous reasoning about probabilities may lead to correct and to wrong results and we should really trust only rigorous proofs. We also have seen that it is easier to argue about $COUNT$ than about BIN. Rudolph (1997) proves that the evolutionary algorithm with $p_m = 1/n$ converges in average time $O(n \ln n)$ for the function $COUNT$ but he could not analyze the behavior of the evolutionary algorithm for BIN and other separable functions.

3 THE ANALYSIS OF THE $(1+1)$ EVOLUTIONARY ALGORITHM FOR BIN AND MUTATION PROBABILITIES OF $1/n$

We rigorously analyze the behavior of the $(1+1)$ evolutionary algorithm (EA) with mutation probability $p_m = 1/n$ for the function BIN. The lower bound works for all separable functions.

Theorem 4: The $(1+1)$ evolutionary algorithm with mutation probability $p_m = 1/n$ needs on average at least $n \ln n - O(n \ln \ln n)$ steps until it reaches the optimal value of a separable function with positive weights, if it starts at $x = (0, \ldots, 0)$.

Proof: It is necessary that each bit flips at least once. Hence, the average time until each bit has tried to flip at least once is a *lower* bound on the considered average time. If a random variable X takes only positive integers, its mean value can be computed by

$$E(X) = \sum_{1 \le t < \infty} t \cdot \text{Prob}(X = t) = \sum_{1 \le t < \infty} \text{Prob}(X \ge t).$$

Let X be the random variable describing the first point of time where each bit has tried to flip. For each bit the probability that it did not try to flip among the first $t-1$ trials equals $\left(1-\frac{1}{n}\right)^{t-1}$, the probability that it has tried to flip equals $1-\left(1-\frac{1}{n}\right)^{t-1}$. The probability that this happens for all n independent bits equals $\left(1-\left(1-\frac{1}{n}\right)^{t-1}\right)^{n}$. Hence,

$$\text{Prob}(X \ge t) = 1 - \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right)^{n} \quad \text{and}$$

$$\begin{aligned} E(X) &= \sum_{1 \le t < \infty} \left[1 - \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right)^{n}\right] \\ &\ge ((n-1)(\ln n - \ln\ln n)) \left(1 - \left(1 - \left(1 - \frac{1}{n}\right)^{(n-1)(\ln n - \ln\ln n)}\right)^{n}\right). \end{aligned}$$

It is well known that $\left(1-\frac{1}{n}\right)^{n} \le e^{-1} \le \left(1-\frac{1}{n}\right)^{n-1}$. Hence,

$$\left(1 - \frac{1}{n}\right)^{(n-1)(\ln n - \ln\ln n)} \ge e^{-(\ln n - \ln\ln n)} = \frac{\ln n}{n} \quad \text{and}$$

$$\left(1 - \left(1 - \frac{1}{n}\right)^{(n-1)(\ln n - \ln\ln n)}\right)^{n} \le \left(1 - \frac{\ln n}{n}\right)^{n} \le e^{-\ln n} = \frac{1}{n}.$$

Altogether,

$$\begin{aligned} E(X) &\ge (n-1)(\ln n - \ln\ln n)\left(1 - \frac{1}{n}\right) \\ &= n \ln n - O(n \ln\ln n) \end{aligned}$$

□

Our estimations are quite tight. Indeed, the probability that $X \ge n(\ln n + \ln\ln n)$ is approximately $1/\ln n$ and the probability that $X \ge 2n \ln n$ only $1/n$. But Theorem 4 says nothing about an upper bound. There are two reasons which slow down the convergence process. First, bits may try to flip from 0 to 1 but the total weight of bits which at the same time try to flip from 1 to 0 is larger than the total weight of the bits which try to flip from 0 to 1. Then the new vector x' is not accepted. Second, bits can flip from 1 to 0 (and afterwards have to flip again from 0 to 1) in a successful step, since the total weight of the bits which try to flip from 0 to 1 is at least of the same size as the total weight of the bits which try to flip from 1 to 0. Altogether, there is no trivial good upper bound on the average time until Algorithm 2 reaches the optimal vector.

Theorem 5: The $(1+1)$ evolutionary algorithm with mutation probability $p_m = 1/n$ needs for BIN on average not more than $2\left(e + e^{1/2}\right) n \ln n \leq 8.8 n \ln n$ steps to reach the optimal vector $(1, \ldots, 1)$.

Proof: Our analysis is independent from the initial vector. Let T be the random variable which describes the point of time where the evolutionary algorithm reaches the optimal vector $x^* = (1, \ldots, 1)$ for the first time. W.l.o.g. n is even. Then $T = T_1 + T_2$ where the random variable T_1 describes the first point of time when $n/2$ leading bits of x all are 1 and $T_2 = T - T_1$ describes the remaining time until x^* is reached. Because of the definition of BIN leading ones never are replaced by zeros.

To estimate $E(T_1)$ we distinguish between successful steps (the new vector x' is accepted and one of the leading $n/2$ bits has changed) and the other unsuccessful steps (which only may change the lower $n/2$ bits if the new vector x' is accepted). Let X_i be the random variable describing the first point of time where the first half of x contains at least i ones, in particular $X_0 = 0$. Then

$$T_1 = X_{n/2} = (X_1 - X_0) + (X_2 - X_1) + \cdots + (X_{n/2} - X_{n/2-1}).$$

Let Y_i be the random number of successful steps among the points of time $X_{i-1}+1, \ldots, X_i$ and Z_i the random number of unsuccessful steps. Then

$$T_1 = Y_1 + Z_1 + Y_2 + Z_2 + \cdots + Y_{n/2} + Z_{n/2}.$$

First, we investigate Y_i. For a successful step it is known that the leftmost flipping bit flips from 0 to 1. The distribution of the random number of further flipping bits in the left half is a binomial distribution with parameters $n/2 - j < n/2$ and $1/n$, if x_j is the leftmost flipping bit. In each case the average number of further flipping bits is less than $1/2$. In order to get an upper bound on Y_i we pessimistically assume that all other flipping bits flip from 1 to 0. Let D_k describe the random difference in the number of ones among the leading $n/2$ bits of x after and before the k-th successful step among the steps $X_{i-1}+1, \ldots, X_i$. By our assumptions, $D_k \leq 1$ and $E(D_k) \geq 1/2$. Let $S_k = D_1 + \cdots + D_k$. Then Y_i is bounded above by S^*, the smallest index k where $S_k = 1$. The process $S_0, S_1, S_2, \ldots$ is a random walk on the line starting at $S_0 = 0$. In the first step we move to $S_1 = D_1$, then to $S_1 + D_2 = S_2$, and so on. Since $D_k \leq 1$, we reach 1 as first point to the right of 0. Then the process stops and we are interested in the average stopping time $E(S^*)$.

Our random walk is homogeneous with respect to time and place. Let $D_1 = d$. Then the distance to 1 equals $1 - d$. Besides the first step we have to wait on average $(1 - d)E(S^*)$ further successful steps to reach 1 for the first time (Theorem on Conditional Expectations). Hence,

$$\begin{aligned}
E(S^*) &= 1 + \sum_{-n/2+2 \leq d \leq 1} \mathrm{Prob}(D_1 = d)(1-d)E(S^*) \\
&= 1 + E(S^*) - E(S^*) \sum_{-n/2+2 \leq d \leq 1} d \cdot \mathrm{Prob}(D_1 = d) \\
&= 1 + E(S^*) - E(S^*)E(D_1).
\end{aligned}$$

Now we can conclude that

$$E(S^*) = 1/E(D_1) \leq 2.$$

The last inequality on $E(D_1)$ has been proven above. The equality $E(S^*)E(D_1) = 1$ is known in more general form in probability theory as Wald's equality. Altogether $E(Y_i) \leq 2$ for all i.

If the probability that a step is successful equals p, the average time until we have k successful steps equals k/p (mean of the negative binomial distribution). Again we work with conditional probabilities. Then

$$\begin{aligned} E(Y_i + Z_i) &= \sum_k \text{Prob}(Y_i = k) E(Y_i + Z_i | Y_i = k) \\ &= \sum_k \text{Prob}(Y_i = k) k/p = E(Y_i)/p \leq 2/p. \end{aligned}$$

Hence, we look for a lower bound on the probability that a step is successful. During the steps $X_i + 1, \ldots, X_{i+1}$ the number of ones in the left half of x is bounded above by i. A sufficient condition for a step to be successful is that all bits equal to 1 do not try to flip and exactly one of the bits equal to 0 tries to flip. The probability of this event equals, if $r \leq i \leq n/2 - 1$ is the actual number of ones,

$$\binom{n/2 - r}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n/2-1} = \frac{1}{n}(n/2 - r)\left(1 - \frac{1}{n}\right)^{(n-2)/2} \geq \frac{1}{n} e^{-1/2}(n/2 - i).$$

Altogether we have

$$\sum_{1 \leq i \leq n/2} E(Y_i + Z_i) \leq 2 \sum_{0 \leq i \leq n/2-1} \left(\frac{1}{n} e^{-1/2}\,(n/2 - i)\right)^{-1} = 2e^{1/2} n \sum_{1 \leq i \leq n/2} \frac{1}{i} \leq 2e^{1/2} n \ln n.$$

The last estimation of the harmonic series follows from the estimation $1 + \frac{1}{2} + \cdots + \frac{1}{n/2} \leq \ln(n/2) + \gamma$ for the Eulerian constant $\gamma \leq 0.58$ and the fact that $\ln(1/2) = -\ln 2 \leq -0.69$.

Finally, we have to investigate the second phase. The analysis of the number of successful steps is the same as before. In the analysis of the probability of a successful step we have to replace $\left(1 - \frac{1}{n}\right)^{n/2-1}$ by $\left(1 - \frac{1}{n}\right)^{n-1}$ to guarantee that no bit of the left half tries to flip. With $\left(1 - \frac{1}{n}\right)^{n-1} \geq e^{-1}$ we get the upper bound $2en \ln n$ on the average time of the second phase. Summing up the upper bounds on $E(T_1)$ and $E(T_2)$ we obtain the desired bound. □

4 THE ANALYSIS OF OTHER MUTATION RATES

In Section 3 we have analyzed the $(1 + 1)$ evolutionary algorithm for the most often recommended mutation probability $p_m = 1/n$. What happens if we decrease or increase this mutation probability? If $p_m = c/n$ for some constant c, we still can prove that the average run time is $\Theta(n \ln n)$. The proof of Theorem 4 can be done in the same way and for the upper bound it is sufficient to partition x to $\lceil c \rceil + 1$ parts of equal size. But we might expect that the behavior becomes worse if the mutation probability changes a lot. If it becomes too small, we have to wait too long until all bits have tried to flip. If it becomes too large, the probability of successful steps becomes too small, since a lot of bits try to flip at the same time. We rigorously prove that these ideas are correct.

Theorem 6: The $(1+1)$ evolutionary algorithm with mutation probability $p_m = \frac{1}{n\alpha(n)}$, where $\alpha(n) \to \infty$ for $n \to \infty$, needs on average $\Omega(\alpha(n) n \ln n)$ steps until it reaches the optimal value of a separable function with positive weights, if it starts at $x = (0, \ldots, 0)$.

Proof: Like in the proof of Theorem 4 let X be the random variable describing the first point of time where each bit has tried to flip. If $\text{Prob}(X \geq t) \geq c$ for some constant c and all $t \leq \alpha(n) n \ln n - \ln n$, then the theorem follows. Let $t = (n\alpha(n) - 1) \ln n$. Then

$$\left(1 - \frac{1}{n\alpha(n)}\right)^t \geq e^{-\ln n} = \frac{1}{n} \quad \text{and}$$

$$\text{Prob}(X \geq t+1) \;=\; 1 - \left(1 - \left(1 - \frac{1}{n\alpha(n)}\right)^t\right)^n \geq 1 - \left(1 - \frac{1}{n}\right)^n \geq 1 - e^{-1}.$$

□

Theorem 7: The (1+1) evolutionary algorithm with mutation probability $p_m = \alpha(n)/n$, where $\alpha(n) \to \infty$ for $n \to \infty$, needs on average $e^{\Omega(\alpha(n))} n \ln n = \Omega(\alpha(n) n \ln n)$ steps until it reaches for BIN the optimal value, if it starts at $x = (x_1, \ldots, x_n)$ where $x_i = 1$, if $i \leq n/2$, and $x_i = 0$, if $i > n/2$.

Proof: Because of the definition of BIN a step only can be successful, if none of the leading $n/2$ bits tries to flip. The probability that a step is successful is, therefore, bounded above by $\left(1 - \frac{\alpha(n)}{n}\right)^{n/2} = e^{-\Omega(\alpha(n))}$. We prove that the average number of successful steps is $\Omega(n(\ln n)/\alpha(n))$. Hence, the average total run time is $\Omega\left((e^{\Omega(\alpha(n))} n \ln n)/\alpha(n)\right) = e^{\Omega(\alpha(n))} n \ln n$.

For the lower bound on the number of successful steps it is sufficient to remark that each of the $n/2$ not leading bits has to flip in at least one successful step. Similarly to the proof of Theorem 6 we choose $t = (n/\alpha(n) - 1) \ln n$. Then

$$\left(1 - \frac{\alpha(n)}{n}\right)^t \geq \frac{1}{n} \quad \text{and}$$

$$\text{Prob}(X \geq t+1) \;=\; 1 - \left(1 - \left(1 - \frac{\alpha(n)}{n}\right)^t\right)^{n/2} \geq 1 - \left(1 - \frac{1}{n}\right)^{n/2} \geq 1 - e^{-1/2}.$$

Since we only are considering $n/2$ bits the outer exponent is here $n/2$ instead of n leading to the constant $1 - e^{-1/2}$ instead of $1 - e^{-1}$. □

5 THE (1+1) EVOLUTIONARY ALGORITHM FOR OTHER SEPARABLE FUNCTIONS

Because of the results of Section 4 we only consider the case of mutation probabilities $p_m = c/n$ for some constant c. Is it possible to use the technique of the proof of Theorem 5? There we could assume that in the second phase the $n/2$ leading bits stay fixed on their value 1. This argument relies heavily on the special definition of BIN. But, if $c < 1$, we can get rid of the second phase and can use the ideas of the proof of Theorem 5. Surprisingly, an analysis is easier for $c < 1$ than for the usual case $c = 1$.

Theorem 8: The $(1+1)$ evolutionary algorithm with mutation probability $p_m = c/n$ for some $c < 1$ needs for separable functions on average $O(n \ln n)$ steps to reach the optimal vector.

Proof: We start as in the proof of Theorem 5 but consider only one phase. Then

$$T = Y_1 + Z_1 + Y_2 + Z_2 + \cdots + Y_n + Z_n.$$

First, we investigate Y_i. If a step is successful, then at least one bit flips from 0 to 1. Pessimistically, we assume that no other bits flip from 0 to 1. The random number of bits which try to flip from 1 to 0 is not larger than given by the binomial distribution with parameters n and c/n. Hence, for D_k (see the proof of Theorem 5) we get $D_k \le 1$ and $E(D_k) \ge 1 - n(c/n) = 1 - c$ which is a constant larger than 0. We conclude by Wald's equality that $E(S^*) = 1/E(D_1) \le (1-c)^{-1}$. Hence, $E(Y_i + Z_i) \le (1-c)^{-1}/p_i$, where p_i is a lower bound on the probability that a step is successful before the process produces a vector with i ones. Similarly to the proof of Theorem 5 we obtain for some positive constants c_1 and c_2 the following estimations.

$$\begin{aligned} p_{i+1} &\ge \binom{n-i}{1} \frac{c}{n} \left(1 - \frac{c}{n}\right)^{n-1} \\ &= c\left(1 - \frac{c}{n}\right)^{n-1} \frac{n-i}{n} \ge c_1 \frac{n-i}{n} \quad \text{and} \end{aligned}$$

$$\sum_{1 \le i \le n} E(Y_i + Z_i) \le c_2 n \sum_{1 \le i \le n} \frac{1}{i} = O(n \ln n).$$

□

It would be annoying if we could not cope with the case $c = 1$.

Theorem 9: The $(1+1)$ evolutionary algorithm with mutation probability $p_m = 1/n$ needs for separable functions on average $O(n \ln n)$ steps to reach the optimal vector.

Proof: The analysis of the first phase of the process until the number of ones reaches for the first time αn for some constant $\alpha < 1$ follows the lines of the proofs of Theorem 5 and Theorem 8. We only have to remark that the number of bits flipping from 1 to

0 in successful steps can be estimated above by a random variable which is binomially distributed with parameters αn and $1/n$. Hence, $D_k \leq 1$ and $E(D_k) \geq 1-\alpha n(1/n) = 1-\alpha$ which is a constant larger than 0. For our purpose $\alpha = 9/10$ is appropriate.

Up to now we have used the number of ones in x as "value" of x and we have measured the progress of the evolutionary algorithm by this value instead of the fitness function f. Sometimes it was useful to distinguish between the $n/2$ leading bits and the other $n/2$ bits. Here we use a mixture of these two approaches. Let

$$val(x) := 2 \sum_{1\leq i\leq n/2} x_i + \sum_{n/2<i\leq n} x_i.$$

Remember that the weights are sorted. Hence, we believe that a one among the leading bits is more "important" than a one among the other bits. Moreover, the value function val is simple enough to be analyzed.

Again, we first investigate successful steps and measure the progress with respect to val. We define a random walk on the line such that the random variable describing this random walk grows slower than the function val for the evolutionary algorithm. Let us consider a successful step and let x_i be the leftmost bit which flips from 0 to 1.

Case 1: $i \leq n/2$. Pessimistically, we assume that no other bit flips from 0 to 1 and that the bits flipping from 1 to 0 are chosen independently with probability $1/n$. Because $f(x') \geq f(x)$ for successful steps we are overestimating the number of new zeros. We like to apply again Wald's equality. To ensure that val increases at most by 1 we assume that, if by the described process no bit flips from 1 to 0, we choose a bit from the second half and let it flip from 1 to 0. What do we know about $val(x') - val(x)$? This difference increases by 2, since x_i flips from 0 to 1. The random number of bits of the first (resp. second) half flipping from 1 to 0 is bounded above by a random variable which is binomially distributed with parameters $n/2$ and $1/n$. Hence, in both cases the average number of bits is bounded by $1/2$. Under this assumption the probability that no bit flips from 1 to 0 is $\left(1-\frac{1}{n}\right)^n$ or approximately e^{-1}. Hence, we overestimate the average number of bits of the first half which flip from 1 to 0 by $1/2$ and for the second half by $1/2+e^{-1}$. Hence,

$$E\left(val(x') - val(x)\right) \geq 2 - \frac{1}{2}\cdot 2 - \left(\frac{1}{2}+e^{-1}\right)\cdot 1 > \frac{1}{10}$$

is bounded below by a positive constant.

Case 2: $i > n/2$. By assumption, all bits flipping from 0 to 1 are in the second half of x. Let k be the number of zeros in x. For a successful step we have $f(x') \geq f(x)$. Since the weights are sorted, each bit flipping from 0 to 1 allows at most one bit in the first half flipping from 1 to 0. Perhaps all other bits of the second half may flip from 1 to 0. Hence, the worst case with respect to val is the case where $x_{n/2+1} = \cdots = x_{n/2+k} = 0$, $w_1 = \cdots w_{n/2} = \cdots = w_{n/2+k} = n$, and $w_{n/2+k+1} = \cdots = w_n = 1$. Let j be the number of bits flipping from 0 to 1. Then the following bounds hold on the number of bits flipping from 1 to 0:

– at most j bits from the first half can flip but no bit from the second half.

– at most $j-1$ bits from the first half can flip and all bits from the second half.

We work under the assumption that $x_{n/2+1}$ flips from 0 to 1. Let B_1 be the random number of bits of the first half which flip from 1 to 0, B_2 the random number of bits among $x_{n/2+2}, \ldots, x_{n/2+k}$ which flip from 0 to 1, and let B_3 be the random number of bits among $x_{n/2+k+1}, \ldots, x_n$ which flip from 1 to 0.

By the considerations at the beginning of the proof we can assume that $k \leq n/10$. Hence,

$$E(val(x') - val(x)) = \sum_{0 \leq j \leq n/10} \text{Prob}(B_2 = j) E(val(x') - val(x) | B_2 = j).$$

Now we distinguish the both scenarios considered above. In the first case

$$E(val(x') - val(x) | B_2 = j) = j + 1 - 2E(B_1 | B_1 \leq j + 1)$$

and in the second case

$$E(val(x') - val(x) | B_2 = j) = j + 1 - 2E(B_1 | B_1 \leq j) - E(B_3).$$

It is obvious that $E(B_3) \leq 1/2$. Furthermore, $E(B_1 | B_1 \leq j) \leq E(B_1 | B_1 \leq j + 1) \leq E(B_1) \leq 1/2$, $E(B_1 | B_1 \leq 0) = 0$, and

$$E(B_1 | B_1 \leq 1) = \text{Prob}(B_1 = 1 | B_1 \leq 1) = \text{Prob}(B_1 = 1) / \text{Prob}(B_1 \leq 1).$$

We know that

$$\begin{aligned}
\text{Prob}(B_1 = 0) &= \left(1 - \frac{1}{n}\right)^{n/2} \quad \text{and} \\
\text{Prob}(B_1 = 1) &= \frac{n}{2} \cdot \frac{1}{n} \cdot \left(1 - \frac{1}{n}\right)^{n/2-1} = \frac{1}{2}\left(1 - \frac{1}{n}\right)^{n/2-1}.
\end{aligned}$$

Altogether

$$E(B_1 | B_1 \leq 1) = \frac{\frac{1}{2}\left(1 - \frac{1}{n}\right)^{n/2-1}}{\left(1 - \frac{1}{n}\right)^{n/2} + \frac{1}{2}\left(1 - \frac{1}{n}\right)^{n/2-1}} = \frac{1}{2\left(1 - \frac{1}{n}\right) + 1} \leq 0.43,$$

if $n \geq 3$. Now for the first case

$$\begin{aligned}
E(val(x') - val(x)) &\geq \text{Prob}(B_2 = 0)(1 - 2 \cdot 0.43) + \sum_{1 \leq j \leq n/10} \text{Prob}(B_2 = j) \cdot 0 \\
&= 0.14 \text{Prob}(B_2 = 0) \geq 0.14 e^{-1/10} > 0.1
\end{aligned}$$

and for the second case

$$\begin{aligned}
E(val(x') - val(x)) &\geq \text{Prob}(B_2 = 0) \cdot (1 - 0.5) \\
&\quad + \sum_{1 \leq j \leq n/10} \text{Prob}(B_2 = j)\,(j + 1 - 2E(B_1 | B_1 \leq j) - 0.5) \\
&\geq e^{-1/10} + 0 - 0.5 > 0.1.
\end{aligned}$$

Now, by Wald's equality the average time that our random walk starting at some point $a = val(x)$ reaches for the first time $a + 1$ is a constant. Starting with at least $(9/10)n$ ones the "value" has to increase by at most $(2/10)n$. The probability of success can be estimated as before by the probability that no 1 tries to flip and at least one 0 tries to flip. If the value function is d away from the maximal value $(3/2)n$, there are at least $d/2$ zeros in x. Hence, we have to consider each of the cases of k zeros, $0 \leq k \leq n$, twice. This leads to a further factor 2 and altogether to the upper bound $O(n \ln n)$. □

6 ON MORE GENERAL EVOLUTIONARY ALGORITHMS

First, we discuss the parallel evolution of a lot of species. This does not help much. If we consider the mutation probability $p_m = 1/n$, our standard calculations show that the probability that all bits have tried to flip within $\frac{1}{2} n \ln n$ steps is bounded by $e^{-n^{1/2}}$ which is exponentially small. Even if we evolve polynomially many species the probability of a successful one is exponentially small.

Another possibility is the use of the crossover operator. We consider uniform crossover and can refer to Mühlenbein and Schlierkamp-Voosen (1993a, 1993b) and make only some remarks. Let x and x' be two vectors of length n. Since there are no variables building natural groups, it is natural to create two children y and y' in the following way. The bit y_i equals x_i with probability 1/2 and x'_i with probability 1/2, y'_i is equal to the remaining bit. If $x_i = x'_i$, also y_i and y'_i get this value. Let k be the number of indices i where $x_i \neq x'_i$. Then the number of ones in y among these positions is binomially distributed with parameters k and 1/2. The average value is $k/2$ and by Chernoff's bounds the value is very close to $k/2$ with very high probability. Hence, the children tend to be of equal quality and for separable functions the sum of their fitness values is equal to the sum of the fitness values of their parents. If we choose among two parents and their two children the two fittest ones this is one parent and one child. If the parents are almost of the same quality, then so are the children. Otherwise, the fitter child typically is not fitter than the fitter parent. A more detailed analysis can prove that the $(1 + 1)$ evolutionary algorithm is adequate for separable functions.

CONCLUSION

We have presented probability theoretical tools to rigorously analyze evolutionary algorithms. We have shown how one can define probabilistic models, in particular random walks, which are easier to analyze than evolutionary algorithms and which are defined in such a way that results on these models imply results on evolutionary algorithms. We rigorously have proved that the $(1 + 1)$ evolutionary algorithm needs $O(n \ln n)$ steps for separable functions and the mutation probability $p_m = 1/n$. We also have proved that much larger or smaller mutation probabilities are worse than the typically used probability of $1/n$.

References

Bäck, T. (1993). Optimal mutation rates in genetic search. Proc. of Fifth Int. Conf. on Genetic Algorithms (S. Forrest, ed.). Morgan Kaufman, San Mateo, CA, 2–8.

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

Feller, W. (1968). An Introduction to Probability Theory and Its Applications. Wiley.

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.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1993a). Predictive models for the breeder genetic algorithm I. Evol. Comput. 1, 25–50.

Mühlenbein, H. and Schlierkamp-Voosen, D. (1993b). The science of breeding and its application to the breeder genetic algorithm. Evol. Comput. 1, 335–360.

Rechenberg, I. (1994). Evolutionsstrategie '94. Frommann-Holzboog.

Rudolph, G. (1997). Convergence Properties of Evolutionary Algorithms. Ph. D. thesis. Verlag Dr. Kovač, Hamburg.

Salomon, R. (1996). Re-evaluating genetic algorithm performance under coordinate rotation of benchmark functions. A survey of some theoretical and practical aspects of genetic algorithms. Bio Systems 39, 263–278.

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

Sedgewick, R. (1991). Algorithms. Addison-Wesley.

van Leeuwen, J. (1990) (Ed.). Handbook of Theoretical Computer Science. Elsevier, MIT Press.