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

Analysis of the (1+1) EA for a Dynamically Changing Objective Function

Stefan Droste

No. CI-113/01

Technical Report ISSN 1433-3325 Juni 2001

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.

Analysis of the (1+1) EA for a Dynamically Changing Objective Function

Stefan Droste
LS Informatik 2, Universität Dortmund
44221 Dortmund, Germany

Abstract

Evolutionary algorithms (EAs) are a class of randomized search heuristics, that are often successfully used for black-box optimization. Nevertheless, there are only few theoretical results about EAs, which are furthermore limited to static objective functions, i. e. functions that do not change over time, despite of the practical relevance of dynamic optimization. Here, the runtime of a simple EA, the (1+1) EA, is theoretically analyzed for a dynamically changing objective function. The main focus lies on determining the degree of change of the fitness funcion, where the expected runtime of the (1+1) EA changes from polynomially to super-polynomially. The proofs presented show methods how to analyze EAs with dynamically changing objective functions.

1 Introduction

Evolutionary algorithm (EA) is a synonym for a randomized search heuristic, whose basic principles, like mutation, recombination, and selection, are borrowed from theories about natural evolution. The main classes of EAs are genetic algorithms (Goldberg (1989)), evolution strategies (Schwefel (1995)), and genetic programming (Koza (1992)). All three classes of EAs are used successfully in practical applications (see Bäck, Fogel, and Michalewicz (1997) for an overview of EAs and their applications).

EAs can be used even if there is no knowledge about the structure of the objective function $f : A \mapsto B$ to be optimized apart from the possibility to compute $f(a)$ for arbitrary $a \in A$. It is intuitively clear, that additional knowledge about the objective function is necessary in order to design more efficient EAs. Formally this was proven by the so-called "No-Free-Lunch"-Theorem (Wolpert and Macready (1997)), stating that on average over all functions all search algorithms perform the same. Hence, theoretical investigations have to concentrate on clearly defined rather small classes of functions.

Because there is no well-laid basis for the theoretical analysis of EAs, we cannot expect to analyze EAs used in practice, which often use rather involved genetic operators, being tailored on the problem at hand. Instead, the first theoretical results focus on the analysis of simple mutation-based evolutionary algorithms, where the so-called (1+1) EA has probably gained the most attention (see Rudolph (1997) or Droste, Jansen, and Wegener (2001), for instance). Although these results are not directly applicable to practical EAs, they show how to analyze EAs, which led to more sophisticated methods making it possible to analyze more and more complicated and "realistic" EAs (see Jansen and Wegener (1999) or Jansen and Wegener (2001)).

All these theoretical results investigate the case where the objective function is static, i. e. does not vary over time. Although dynamically changing objective functions have wide practical applications (see Bryson (1998), investigations of EAs hereon are limited to experimental work or heuristic approximative analysis (see Sarma and DeJong (1999), Ronnewinkel and Wilke (2001), or Weicker and Weicker (1999)). Here we rigorously analyze the runtime, i. e. the number of individuals that are evaluated until an optimum of the actual objective function is found, of the (1+1) EA for a dynamically changing objective function. Although the algorithm and the objective function are very simple with respect to algorithms and problems in practice, the paper shows how dynamic objective functions can be theoretically analyzed.

The paper is structured as follows: in the next section we formally define the (1+1) EA for dynamic objective functions $f_t : \{0,1\}^n \mapsto \mathbb{R}$ and its runtime. We also introduce our model of a randomized process changing the objective function: we measure the fitness as the number of matching positions in the actual individual and an objective bitstring, where one uniformly chosen bit is changed in the objective bitstring with probability p_y during one generation. In Section 3 we show that for $p_y = O(\log(n)/n)$ the expected runtime of the (1+1) EA is polynomially in n, and in Section 4 that for every substantially larger probability, i. e. $p_y = \omega(\log(n)/n)$, the expected runtime becomes super-polynomially. The paper ends with some conclusions.

2 The (1+1) EA for dynamically changing objective functions

The (1+1) EA is the most simple EA for maximizing an objective function $f : \{0,1\}^n \mapsto \mathbb{R}$: its population consists of only one individual $x \in \{0,1\}^n$, which is uniformly chosen during initialization. In every generation the actual bitstring is mutated to a bitstring $x' \in \{0,1\}^n$ by randomly flipping each bit of x with probability $p_x = 1/n$. If the fitness $f(x')$ of the new bitstring is at least as large as that of the old bitstring x, the bitstring x' is chosen as successor, i. e. x is replaced by x'. In practice, this mutation-selection process will be repeated until a stopping criterion becomes valid. In theoretical investigations we are interested in the number of generations, until an optimum is found for the first time, hence we omit any stopping criterion.

We model a dynamically changing objective function as a family $(f_t : \{0,1\}^n \to \mathbb{R})_{t\in\mathbb{N}_0}$ of functions, where f_t is the actual objective function after the (1+1) EA has made $t \in \mathbb{N}_0$ steps. Hence, also the actual individual is denoted as x_t to make the notation clearer. The formal definition of the (1+1) EA is now as follows:

Algorithm 2.1 *(1+1) EA for a dynamically changing objective function* $(f_t : \{0,1\}^n \mapsto \mathbb{R})_{t\in\mathbb{N}_0}$.

1. *Set* $t := 0$ *and choose* $x_t \in \{0,1\}^n$ *randomly.*
2. *Set* $x' := x_t$ *and flip each bit of* x' *independently with probability* $1/n$.
3. *If* $f_t(x') \geq f_t(x_t)$, *set* $x_{t+1} := x'$, *else* $x_{t+1} := x_t$.
4. *Set* $t := t + 1$ *and go to step 2.*

Hence, after generating x' by mutating x_t the actual objective function is f_t. So it makes sense to use f_t instead of the "old" f_{t-1} to compare x' and x_t, because it is our goal to find an element x_t being optimal for the actual objective function f_t at this time. We examine the runtime T_f, where f denotes the family of objective functions to be optimized:

Definition 2.2 *Let* $f = (f_t : \{0,1\}^n \mapsto \mathbb{R})_{t\in\mathbb{N}_0}$ *be a family of objective functions. The* runtime T_f *of the (1+1) EA is defined as*

$$T_f := \min\{t \in \mathbb{N}_0 \mid f_t(x_t) = \max\{f_t(x) \mid x \in \{0,1\}^n\}\}.$$

Hence, the runtime is the first point in time, where the (1+1) EA reaches an optimum of the actual objective function. Surely, this is just one aspect of the behaviour of the (1+1) EA: another important aspect for dynamically changing objective functions is the degree to which a found optimum can be lost, after it has been found.

One of the first functions the (1+1) EA has been analyzed for is OneMax, where its expected runtime is $O(n \log(n))$ (see (Mühlenbein (1992) for a first approximate analysis and (Droste, Jansen, and Wegener (1998) for a rigorous analysis). OneMax returns as its function value the number of ones in its argument, which can also be interpreted as the number of bits, where the argument matches an objective bitstring, the all-one bitstring $(1, \ldots, 1)$. Here we look at a dynamic variant called $\text{OneMax}_{t,M}$, where the objective bitstring $(1, \ldots, 1)$ is changed by a random operator $M : \{0,1\}^n \to \{0,1\}^n$ (formally it should be a function $M : \{0,1\}^n \times \Omega \mapsto \{0,1\}^n$, where Ω forms a probability space together with a probability measure P, but we will omit these details for the sake of brevity):

Definition 2.3 *Let the family* $(\text{OneMax}_{t,M} : \{0,1\}^n \mapsto \{0,1\})_{t \in \mathbb{N}}$ *of functions be defined as*

$$\text{OneMax}_{t,M}(x) := n - H(x, y_t), \text{ where } y_0 := (1, \ldots, 1) \text{ and } y_{t+1} := M(y_t).$$

Here, $H(x, y)$ is the Hamming distance between x and y. Because of symmetry reasons it makes no difference to initialize y_0 with another bitstring than $(1, \ldots, 1)$ with respect to the runtime of the (1+1) EA. There are many choices for the random operator M changing the objective bitstring y_t. We look at one of the simplest, where M flips exactly one uniformly chosen bit of the argument with probability p_y and does not change the argument in the other case:

Definition 2.4 *Let* $p_y \in [0,1]$*. Then the random operator* $M_1 : \{0,1\}^n \mapsto \{0,1\}^n$ *is defined as:*

$$P(M_1(y) = y') = \begin{cases} \frac{p_y}{n} & , \text{ if } H(y, y') = 1, \\ 1 - p_y & , \text{ if } H(y, y') = 0, \\ 0 & , \text{ if } H(y, y') > 1. \end{cases}$$

Hence, the movement rate of the objective bitstring depends on the objective mutation probability p_y, but in one step only one bit of y_t can flip. Using this random operator M_1, the family of objective functions OneMax_{t,M_1} for $t \in \mathbb{N}_0$ is called OneMax_D for short. In the following section we will show that for $p_y = O(n/\log(n))$ the expected running time of the (1+1) EA with $p_x = 1/n$ on OneMax_D is polynomial.

3 A polynomial upper bound on the expected runtime for small objective mutation rates

In this section, we show that for objective mutation probabilities p_y of size $O(\log(n)/n)$ the (1+1) EA needs polynomial runtime in the expected case. In order to do this, we model a run of the (1+1) EA by a stochastic process, replace this process by a "slower" one, and show that even this process needs only polynomial expected time to reach a state corresponding to the optimum. Because the stochastic process modelling the (1+1) EA will be used in the next section, too, we introduce it here in more detail.

Let the actual search point at the beginning of the t-th iteration of the (1+1) EA be x_t. To make our formalization easier, we assume that the objective bitstring is always $(1, \dots, 1)$, i. e. the fitness of a bitstring is equal to the number of its ones. Therefore, the random operator M_1 originally working on the objective bitstring is "transferred" to the actual search point x_t: with probability p_y one of its bits is chosen randomly and flipped; with probability $1 - p_y$ nothing is done. Let x'_t be its outcome. Then bit-wise mutation is applied with probability p_x per bit on x'_t and results in x''_t.

The selection in step 3 of the (1+1) EA now implicitly decides between x'_t and x''_t: if $|x''_t| \geq |x'_t|$, then $x_{t+1} := x''_t$, otherwise $x_{t+1} := x'_t$. Hence, $|x_{t+1}|$ can be smaller than $|x_t|$ but only by one. Now we have a model with two mutation steps, where the first is always accepted, i. e. influences the search process. The second, bit-wise mutation is only accepted if it does not decrease the number of ones with respect to x'_t. Using this somewhat simpler, equivalent model, we prove a polynomial upper bound for $p_y = O(n/\log(n))$:

Theorem 3.1 *The expected runtime of the (1+1) EA on* $\textsc{OneMax}_D$ *with* $p_y \leq c \cdot \log(n)/n$ *(for a constant* $c > 0$*) is* $O(n^{\frac{c\exp(1)}{\ln(2)} \cdot (1+o(1))+1} \cdot \log(n))$.

Proof: In order to upper bound the runtime of the (1+1) EA we analyze a Markov chain on the set $\{0, \dots, n\}$, whose time T to reach state n stochastically dominates the runtime $T_{\textsc{OneMax}_D}$ of the (1+1) EA, i. e. for all $t \in \mathbb{N}_0$ we have $P(T \geq t) \geq P(T_{\textsc{OneMax}_D} \geq t)$. Then it easily follows that the expected value of T is at least as large as $E(T_{\textsc{OneMax}_D})$. The Markov chain on $\{0, \dots, n\}$ looks as follows: the initial state is chosen as $i \in \{0, \dots, n\}$ with probability $\binom{n}{i}/2^n$. This exactly models the random initialization of the (1+1) EA. From a state $i \in \{0, \dots, n\}$ the only possible successor states are $i - 1$ resp. $i + 1$, where the transition probabilities p_i^- resp. p_i^+ are

$$p_i^- = p_y \cdot \frac{i}{n} \quad \text{resp.} \quad p_i^+ = (1 - p_y) \cdot (n - i) \cdot p_x \cdot (1 - p_x)^{n-1}.$$

As p_i^- is an upper bound for the probability, that the number of ones in x_t decreases by one, and p_i^+ a lower bound for the probability, that the number of ones in x_t increases by one, the resulting Markov process is "slower" than the (1+1) EA (here it is essential that the number of ones in the (1+1) EA can decrease by at most one in a single step). It results from the exact Markov process of the (1+1) EA by setting all transition probabilities from a state i to state $j \geq i+2$ to zero, whereas those from a state i to a state $i+1$ resp. to a state $i-1$ are not increased resp. not decreased (the "remaining probabilities" are added to the probability to stay in state i). Hence, one can easily show by induction over $i \in \{0, \ldots, n-1\}$, that the time T_i for the new Markov chain to reach state n starting from i stochastically dominates the time of the (1+1) EA to come from a string with i ones to the string with n ones. As the initialization of the (1+1) EA is exactly modelled by the Markov chain, it is sufficient for us to upper bound the expected time $E(T_i)$ to reach state n starting from state i for the first time.

If T_i^+ denotes the number of steps to reach state $i+1$ for the first time when starting in state i, it is well known (see, for instance, (Droste, Jansen, and Wegener (2000a)), that the following equation holds:

$$E(T_i^+) = \sum_{k=0}^{i} \frac{1}{p_k^+} \cdot \prod_{l=k+1}^{i} \frac{p_l^-}{p_l^+}.$$

Filling in our values for p_i^- and p_i^+, we get that $E(T_i^+)$ equals

$$\begin{aligned}
& \sum_{k=0}^{i} \frac{1}{(1-p_y)(n-k)p_x(1-p_x)^{n-1}} \cdot \prod_{l=k+1}^{i} \frac{p_y \frac{l}{n}}{(1-p_y)(n-l)p_x(1-p_x)^{n-1}} \\
= \; & \sum_{k=0}^{i} \frac{1}{(1-p_y)(n-k)p_x(1-p_x)^{n-1}} \cdot \frac{(p_y)^{i-k} \cdot \frac{i!}{k! \cdot n^{i-k}} \cdot \frac{(n-i-1)!}{(n-k-1)!}}{(1-p_y)^{i-k}(p_x)^{i-k}(1-p_x)^{(n-1)(i-k)}} \\
= \; & \sum_{k=0}^{i} \frac{\binom{i}{k}}{\binom{n-k}{i-k}(n-i)} \cdot \left(\frac{p_y}{n(1-p_y)p_x(1-p_x)^{n-1}}\right)^{i-k} \cdot \frac{1}{(1-p_y)p_x(1-p_x)^{n-1}}
\end{aligned}$$

(In order to upper bound this expression we lower bound $\binom{n-k}{i-k}$ by 1:)

$$\begin{aligned}
\leq \; & \frac{1}{(1-p_y)p_x(1-p_x)^{n-1}} \cdot \frac{1}{n-i} \cdot \sum_{k=0}^{i} \binom{i}{k} \left(\frac{p_y}{n(1-p_y)p_x(1-p_x)^{n-1}}\right)^{i-k} \\
= \; & \frac{1}{(1-p_y)p_x(1-p_x)^{n-1}} \cdot \frac{1}{n-i} \cdot \left(1 + \frac{p_y}{n(1-p_y)p_x(1-p_x)^{n-1}}\right)^{i}
\end{aligned}$$

Using $p_y \le c\log(n)/n$ and $p_x = 1/n$, we can upper bound $E(T_i^+)$ by

$$
\begin{aligned}
&\frac{1}{\left(1-c\frac{\log(n)}{n}\right)\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}} \cdot \frac{1}{n-i} \cdot \left(1+\frac{c\frac{\log(n)}{n}}{n\left(1-c\frac{\log(n)}{n}\right)\frac{1}{n}\left(1-\frac{1}{n}\right)^{n-1}}\right)^i \\
\le\ & \frac{\exp(1)}{\frac{n-c\log(n)}{n}} \cdot \frac{n}{n-i} \cdot \left(1+\frac{c\cdot\log(n)\cdot\exp(1)}{n-c\log(n)}\right)^i \\
\le\ & 2\exp(1)\cdot\frac{n}{n-i}\cdot\left(1+\frac{c\cdot\log(n)\cdot\exp(1)}{n-c\log(n)}\right)^i \qquad \text{(for } n \ge 2c\log(n)\text{)}.
\end{aligned}
$$

In order to use the estimation $(1+1/x)^x \le \exp(1)$, we transform the last expression into:

$$
\begin{aligned}
&2\exp(1)\cdot\frac{n}{n-i}\cdot\left(1+\frac{c\cdot\log(n)\cdot\exp(1)}{n-c\log(n)}\right)^{\frac{n-c\log(n)}{c\log(n)\exp(1)}\cdot\frac{c\log(n)\exp(1)}{n-c\log(n)}\cdot i} \\
\le\ & 2\exp(1)\cdot\frac{n}{n-i}\cdot\exp\left(\frac{c\log(n)\exp(1)}{n-c\log(n)}\cdot i\right) \\
\le\ & 2\exp(1)\cdot\frac{n}{n-i}\cdot n^{\frac{c\exp(1)}{\ln(2)}\cdot\frac{n}{n-c\log(n)}} = \frac{2\exp(1)}{n-i}\cdot n^{\frac{c\exp(1)}{\ln(2)}\cdot(1+o(1))+1}.
\end{aligned}
$$

Hence, by linearity of the expectation and pessimistically assuming that we initialize in $(0,\dots,0)$, we can upper bound $E(T_{\textsc{OneMax}_D})$ by

$$
\sum_{i=0}^{n-1} E(T_i^+) \le 2\exp(1)\cdot n^{\frac{c\exp(1)}{\ln(2)}(1+o(1))+1}\cdot\sum_{i=1}^{n}\frac{1}{i} = O\left(n^{\frac{c\exp(1)}{\ln(2)}\cdot(1+o(1))+1}\cdot\log(n)\right).
$$

□

4 A super-polynomial lower bound for large objective mutation rates

After we have shown in the last section, that the expected number of steps to reach a global optimum of $\textsc{OneMax}_D$ is polynomially bounded for the (1+1) EA and $p_y = O(\log(n)/n)$, we now want to analyze how the runtime changes for an essentially larger rate p_y of change of the objective bitstring. Hence, let $p_y = \omega(\log(n)/n)$. Then we denote p_y by $\alpha(n)\cdot\log(n)/n$, where $\lim_{n\to\infty}\alpha(n)=\infty$, i. e. α grows without bounds.

In order to prove a super-polynomial lower bound on the runtime of the (1+1) EA for $\textsc{OneMax}_D$, we show that the probability of the (1+1) EA to reach the global optimum in t steps is super-polynomially small, i. e. at most $1/\beta(n)$, where $\beta(n)$ grows faster than any polynomial. Therefore, it is super-polynomially unlikely that the optimum is reached after polynomially steps. Furthermore, we can rule out a constant number of events in the random process generated by the (1+1) EA, as long as we show that their probability is super-polynomially small.

Theorem 4.1 *The (1+1) EA finds the optimum of* ONEMAX_D *with* $p_y = \omega(\log(n)/n)$ *in* $p(n)$ *steps with super-polynomially small probability for every polynom* p *in* n.

Proof: A rough outline of the proof is as follows: we concentrate on the phase, where the (1+1) EA has found a bitstring with fitness at least $n - G$, where $G = \log(n)\alpha(n)^{6/10} = o(n)$. Then we show that the probability of decreasing the fitness in a mutation is by a non-constant factor higher than that of increasing it. By showing that a mutation increasing the number of ones by more than $L = \alpha(n)^{1/10}$ is super-polynomially unlikely, we can restrict to mutations increasing the number of ones by at most L. Now, a sequence of mutations of the (1+1) EA starting with $n-G$ ones and ending in the optimum has to contain by a constant factor more increasing mutations than their expected value. Using Chernoff bounds we can show that such a sequence is super-polynomially unlikely.

Our first assumption on the (1+1) EA is to start with a bitstring having at most $n - G$ ones. Because $G = \log(n)\alpha(n)^{6/10}$ is $o(n)$, Chernoff bounds guarantee that the probability of initializing with a bitstring with more than $n - G$ ones is exponentially small. Furthermore, we assume that in one mutation at most $L = \alpha(n)^{1/10}$ zeros flip. The probability of mutating at least L of the at most G zeros can be upper bounded by

$$\binom{G}{L} \cdot \left(\frac{1}{n}\right)^L \leq \left(\frac{G}{n}\right)^L = \left(\frac{\log(n) \cdot \alpha(n)^{6/10}}{n}\right)^L \leq \left(\frac{\log(n)}{n}\right)^{4L/10}.$$

Hence, we can rule out mutations increasing the number of ones by more than a constant and thereby only make an error with super-polynomially small probability, if we concentrate on polynomially many steps. Now we can further assume that every bitstring with less than $n - G$ ones is changed to an arbitrary bitstring with exactly $n-G$ ones. Because the distance to the optimum decreases in one mutation by at most $L = \alpha(n)^{1/10}$ and $L = o(G)$, the expected time to reach the optimum cannot increase when diminishing the distance to the optimum to exactly G.

Under this assumption the number of ones of the current bitstring of the (1+1) EA varies between $n - G$ and n and we are interested in the expected time to reach n ones for the first time. Let I_t be the actual state, i. e. the number of ones of x_t. In order to reach state n, there have to be points in time t_1 and t_2, where I_{t_1} is $n-G$, $I_t \in \{n-G+1,\dots,n-1\}$ for all $t \in \{t_1+1,\dots,t_2-1\}$, and $I_{t_2}=n$. In the following we want to show that for every $t = t_2 - t_1 \in \mathbb{N}$ it is super-polynomially unlikely that such a sequence $I_{t_1},\dots,I_{t_2}$ exists (a similar technique was used in Droste, Jansen, and Wegener (2000b)).

Therefore, we upper resp. lower bound the probability p_i^+ resp. p_i^- that the number of ones increases resp. decreases, when the actual number of ones is i. Because the number of ones decreases, if the mutation of the objective bitstring flips one of the i ones and the bitwise mutation flips no zero, we have

$$p_i^- \geq p_y \cdot \frac{i}{n} \cdot \left(1 - \frac{1}{n}\right)^{n-i} \geq \frac{p_y}{\exp(1)} \cdot \frac{i}{n}.$$

To upper bound p_i^+ we have to be more careful. A mutation step, where the number of ones increases, implies one of the three following conditions:

1. The mutation of the objective string does not change a bit and the bitwise mutation flips at least one of the at most $n-i$ zeros.
2. The mutation of the objective string does change one of the at most $n-i$ zeros.
3. The mutation of the objective string does change one of the at least i ones and at least one of the $n-i$ zeros is flipped during the bitwise mutation (the more restrictive condition of at least two flipping zeros in the bitwise mutation lowers the following bound on p_i^+ only by a constant factor and is therefore omitted for an easier description).

Hence, we can upper bound p_i^+ by

$$\begin{aligned}
& (1-p_y) \cdot \left(1 - \left(1 - \frac{1}{n}\right)^{n-i}\right) + p_y \cdot \left(\frac{n-i}{n} + \frac{i}{n} \cdot \left(1 - \left(1 - \frac{1}{n}\right)^{n-i}\right)\right) \\
\leq\ & (1-p_y) \cdot \left(1 - 1 + \frac{n-i}{n}\right) + p_y \cdot \left(\frac{n-i}{n} + \frac{i}{n} \cdot \left(1 - 1 + \frac{n-i}{n}\right)\right) \\
=\ & \frac{n-i}{n} + p_y \cdot \frac{i \cdot (n-i)}{n^2} = \frac{n-i}{n} \cdot \left(1 + p_y \cdot \frac{i}{n}\right) \leq 2 \cdot \frac{n-i}{n}.
\end{aligned}$$

Using that i is at least $n - G = n - \log(n)\alpha(n)^{6/10}$ and $p_y = \alpha(n) \cdot \log(n)/n$, we get

$$p_i^- \geq \frac{\alpha(n)\frac{\log(n)}{n}}{\exp(1)} \cdot \frac{n - \log(n)\alpha(n)^{6/10}}{n} \text{ and } p_i^+ \leq 2 \cdot \frac{\log(n)\alpha(n)^{6/10}}{n}.$$

Because it is sufficient for us that decreasing the number of ones is by a factor $\alpha(n)^{3/10}$ more likely than increasing them, we roughly bound p_i^- and p_i^+ by

$$p_i^- \geq \frac{1}{2\exp(1)} \cdot \frac{\alpha(n)\log(n)}{n} \text{ and } p_i^+ \leq \frac{1}{2\exp(1)} \cdot \frac{\alpha(n)^{7/10}\log(n)}{n}$$

for all $n \geq n_0$ for a constant $n_0 \in \mathbb{N}$ (where $1/(2\exp(1))$ is an arbitrarily chosen constant smaller than $\exp(-1)$).

Hence, the probability that the next mutation changing the number of ones (called *effective mutation*) increases the number of ones is at most

$$\frac{\frac{1}{2\exp(1)} \cdot \frac{\alpha(n)^{7/10}\log(n)}{n}}{(\alpha(n)+\alpha(n)^{7/10}) \cdot \frac{1}{2\exp(1)} \cdot \frac{\log(n)}{n}} = \frac{\alpha(n)^{7/10}}{\alpha(n)+\alpha(n)^{7/10}} \leq \alpha(n)^{-3/10}.$$

Thereby, we have shown that the expected number of increasing mutations during t effective mutations is at most $t/\alpha(n)^{3/10}$. To be able to get a super-polynomially small bound by Chernoff bounds we show that

1. the expected number of increasing mutations grows significantly faster than $\log(n)$, i. e. $t/\alpha(n)^{3/10} = \omega(\log(n))$ and
2. the number of increasing mutations necessary to reach the optimum is by a constant factor larger than their expected number.

In order to prove these claims we argue that t, the length of the sequence, has to be at least $G/L = \log(n)\alpha(n)^{1/2}$, because every increasing mutation raises the number of ones by at most $L = \alpha(n)^{1/10}$. Hence, $t/\alpha(n)^{3/10} = \omega(\log(n))$, i. e. only sequences of some minimum length are candidates as sequences $I_{t_1}, \ldots, I_{t_2}$. The necessary number of increasing mutations can be lower bounded in the following way: if t^+ denotes the number of increasing and t^- the number of decreasing mutations during $t = t^+ + t^-$ effective mutations, the number of ones can be increased by at most $t^+ \cdot L - t^-$. In order to reach the global optimum this value has to be at least G. This is equivalent to

$$t^+ \cdot L - t^- \geq G \iff t^+ \geq \frac{G+t-t^+}{L} \iff t^+ \geq \frac{G+t}{L+1}.$$

For n large enough we have $\alpha(n)^{1/10}+1 \leq \alpha(n)^{2/10}$ and get

$$\frac{G+t}{L+1} = \frac{\log(n)\alpha(n)^{6/10}+t}{\alpha(n)^{1/10}+1} \geq \frac{t}{\alpha(n)^{2/10}}.$$

This means that in order to reach the global optimum in t steps the number of increasing mutations has to be by a non-constant factor larger than the expected number $t/\alpha(n)^{3/10}$ of increasing mutations. Because there exists an $n_1 \in \mathbb{N}_0$ such that

$$\forall n \geq n_1 : \frac{1}{\alpha(n)^{2/10}} \geq \frac{2}{\alpha(n)^{3/10}},$$

using Chernoff bounds we can upper bound the probability of a sequence $I_{t_1}, \ldots, I_{t_2}$ of length t with the desired properties by

$$\exp\left(-\frac{t}{\alpha(n)^{3/10}} \cdot \frac{4}{3}\right).$$

Because t is at least $\alpha(n)^{1/2}\log(n)$, this can be upper bounded by

$$\exp\left(-\Omega(\alpha(n)^{2/10}\log(n)\right) = n^{-\Omega(\alpha(n)^{2/10})},$$

All in all, we have shown that in order for the (1+1) EA to find the optimum of $\textsc{OneMax}_D$ with $p_y = \omega(\log(n)/n)$ in $t = \mathit{poly}(n)$ steps a super-polynomially unlikely event has to happen. Therefore, the (1+1) EA finds the optimum of $\textsc{OneMax}_D$ after polynomially many steps only with super-polynomially small probability which proves the lemma. □

5 Conclusion

Dynamic optimization is an important task in many practical applications, but theoretical investigations of EAs for dynamically changing objective functions are very rare. Here a simple (1+1) EA is analyzed for a simple objective function measuring the distance to an objective bitstring. Assuming that with probability p_y one uniformly chosen bit of the objective bitstring flips, we show that the expected runtime of the (1+1) EA is polynomial if p_y is $O(\log(n)/n)$. For all substantially larger probabilities, i. e. $p_y = \omega(\log(n)/n)$, it is shown that the runtime becomes super-polynomial with high probability. Although this limiting mutation probability $\Theta(\log(n)/n)$ is of its own interest the main focus lies on the methods used to analyze the (1+1) EA. These can be the basis for the analysis of the (1+1) EA on other dynamically changing objective functions. For instance, it is future research to analyze its behaviour on $\textsc{OneMax}_{t,M}$ for a random operator M modifying the objective bitstring by a bit-wise mutation.

References

Bäck, T., D. B. Fogel, and Z. Michalewicz (Eds.) (1997). *Handbook of Evolutionary Computation.* New York, NY: Institute of Physics Publishing and Oxford University Press.

A. E. Bryson (1998). *Dynamic Optimization.* Reading, MA: Addison-Wesley.

S. Droste, T. Jansen, and I. Wegener (1998). A rigorous complexity analysis of the $(1+1)$ evolutionary algorithm for linear functions with boolean inputs. In *Proceedings of the Third IEEE International Conference on Evolutionary Computation* (*ICEC 1998*), Piscataway, NY, 499–504. IEEE Press.

S. Droste, T. Jansen, and I. Wegener (2000a). Dynamic parameter control in simple evolutionary algorithms. In *Proceedings of the Sixth Foundations of Genetic Algorithms Workshop* (*FOGA 2000*). In print.

S. Droste, T. Jansen, and I. Wegener (2000b). A natural and simple function which is hard for all evolutionary algorithms. In *Proceedings of Third Asia-Pacific Conference on Simulated Evolution and Learning* (*SEAL 2000*), Piscataway, NJ, 2704–2709. IEEE Press.

S. Droste, T. Jansen, and I. Wegener (2001). On the analysis of the $(1+1)$ evolutionary algorithm. Accepted for Theoretical Computer Science.

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

T. Jansen and I. Wegener (1999). On the analysis of evolutionary algorithms – a proof that crossover really can help. In *Proceedings of the Seventh Annual European Symposium on Algorithms* (*ESA 1999*), Berlin, 184–193. Springer. Lecture Notes in Computer Science 1643.

T. Jansen and I. Wegener (2001). Real royal road functions – where crossover provably is essential. Accepted for GECCO 2001.

J. R. Koza (1992). *Genetic Programming.* Cambridge, MA: MIT Press.

H. Mühlenbein (1992). How genetic algorithms really work: I. mutation and hillclimbing. In *Proceedings of the Second Conference on Parallel Problem Solving from Nature* (*PPSN 1992*), Berlin, 15–26. Springer.

C. Ronnewinkel and C. Wilke (2001). Dynamic fitness landscapes: Expansions for small mutation rates. *Physica A*(290), 475–490.

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

J. Sarma and K. DeJong (1999). The behaviour of spatially distributed evolutionary algorithms in non-stationary environments. In *Proceedings of the First Genetic and Evolutionary Computation Conference* (*GECCO 1999*), San Francisco, CA, 572–578. Morgan Kaufmann.

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

K. Weicker and N. Weicker (1999). On evolution strategy optimization in dynamic environments. In *Proceedings of the 1999 Congress on Evolutionary Computation* (*CEC 1999*), Piscataway, NJ, 2039–2046. IEEE Press.

D. H. Wolpert and W. G. Macready (1997). No free lunch theorems for optimization. *IEEE Transactions on Evolutionary Computation 1*(1), 67–82.