ELSEVIER

Theoretical Computer Science 276 (2002) 51–81

Theoretical Computer Science

www.elsevier.com/locate/tcs

# On the analysis of the (1 + 1) evolutionary algorithm ☆

Stefan Droste *, Thomas Jansen, Ingo Wegener

*FB Informatik, LS 2, Universitat Dortmund, 44221 Dortmund, Germany*

Received February 1998; received in revised form February 2001; accepted February 2001
Communicated by M. Paterson

**Abstract**

Many experimental results are reported on all types of Evolutionary Algorithms but only few results have been proved. A step towards a theory on Evolutionary Algorithms, in particular, the so-called (1 + 1) Evolutionary Algorithm, is performed. Linear functions are proved to be optimized in expected time $O(n\ln n)$ but only mutation rates of size $\Theta(1/n)$ can ensure this behavior. For some polynomial of degree 2 the optimization needs exponential time. The same is proved for a unimodal function. Both results were not expected by several other authors. Finally, a hierarchy result is proved. Moreover, methods are presented to analyze the behavior of the (1 + 1) Evolutionary Algorithm. 

## 1. Introduction

Evolutionary Algorithms are a class of search algorithms that are often used as function optimizers for static objective functions. There are a lot of different types of Evolutionary Algorithms, the best known are Evolution Strategies [16, 19], Evolutionary Programming [5], Genetic Algorithms [8, 7], and Genetic Programming [12]. Each type of Evolutionary Algorithm itself is an algorithm paradigm that has many different concrete instances. The field of Evolutionary Algorithms, though the first origins can be found in the early 1960s, is still a quite young and evolving area of mostly practical efforts. The very successful application of Evolutionary Algorithms in very different domains led to strongly practical oriented interests. Today, theory is far behind "experimental knowledge". There are, of course, theoretical investigations about some properties of Evolutionary Algorithms, though rigorous research is hard to find. This

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

* Corresponding author.

*E-mail addresses:* droste@ls2.cs.uni-dortmund.de (S. Droste), jansen@ls2.cs.uni-dortmund.de (T. Jansen), wegener@ls2.cs.uni-dortmund.de (I. Wegener).

0304-3975/02/$ - see front matter 
PII: S0304-3975(01)00182-7

implies that even the best known results are still subject to controversial discussions. To put an end to this, a solid theory has to be build up that starts with simple examples and shows how results can be obtained in a rigorous fashion. The most important questions concerning Evolutionary Algorithms, as long as function optimization is the objective, are how efficiently an Evolutionary Algorithm will optimize a given objective function, which classes of objective functions can be optimized efficiently and which cannot.

In order to make a step towards this goal we investigate the running time behavior of a very simple variant, that is called $(1+1)$ Evolutionary Algorithm ($(1+1)$ EA). As the name "Evolutionary Algorithm" suggests, evolution as it is observed in nature is imitated. It is believed that the repeated process of recombination, mutation, and selection leads to individuals that are increasingly adapted to their environment, i.e., they are assumed to be of increasing fitness. Therefore, in EAs a possible solution to the optimization task is called an *individual* and a set of individuals is called a *population*. From this population in one step or *generation* a subset of *parents* is *selected* according to their function values under the objective function, i.e., their *fitness*. The individuals are *recombined* by application of *crossover* and the resulting individuals are *mutated*. Then some *replacement strategy* is applied to determine the next population that may contain some of the newly generated *children* as well as some of their *parents*. The choice of concrete algorithms to perform selection, crossover, mutation, and replacement as well as the choice of the concrete representation of the individuals offers a great variety of different EAs.

The analysis here concentrates on the most simple variant of an EA that is still of theoretical and practical interest [10, 17]. We restrict the size of the population to just one individual and do not use crossover. Since we assume the objective function to have Boolean inputs we represent the current individual as a bit string. This is the usual choice for Genetic Algorithms. We use a bitwise mutation operator that flips each bit independently of the others with some probability $p_m$ that depends on the length of the bit string. We replace the current bit string by the new one if the fitness of the current bit string is not superior to the fitness of the new string. This replacement strategy is taken from Evolution Strategies and is called $(1+1)$-strategy there [17]: the new generation is chosen as the best individual of one parent and one child. We always assume that the objective or fitness function $f: \{0,1\}^n \to \mathbb{R}$ has to be maximized here. Therefore, we can formalize the $(1+1)$ EA as follows.

**Algorithm 1**

1. *Set* $p_m := 1/n$.
2. *Choose randomly an initial bit string* $x \in \{0,1\}^n$.
3. *Repeat the following mutation step*:
   *Compute* $x'$ *by flipping independently each bit* $x_i$ *with probability* $p_m$.
   *Replace* $x$ *by* $x'$ *iff* $f(x') \geqslant f(x)$.

Algorithm 1 is sometimes regarded as a special variant of a hillclimber and is then said to be a randomized or stochastic hillclimber (cf. e.g. [1]). Like a hillclimber it

uses only one current point in the search space and never accepts a new point with inferior function value. Unlike normal hillclimbers the $(1+1)$ EA has no clearly defined neighborhood, or, stated otherwise, it can reach in one single step any point in the search space, while the probability of reaching a point decreases with increasing Hamming distance to the current point.

One may consider the $(1+1)$ EA as a degenerate kind of Simulated Annealing [11], where the cooling scheme is trivial since the temperature is constant zero. In this case, again, the probabilistic kind of neighborhood is quite unusual.

We start off here with some basic definitions that will be helpful during the analysis. More concepts and notions are introduced in the sections, when they are needed.

**Definition 2.** A function $f:\{0,1\}^n \to \mathbb{R}$ is called a *fitness function*. We assume that $f$ has to be maximized.

**Definition 3.** For two bit strings $x, y \in \{0,1\}^n$ we define the *Hamming distance* of $x$ and $y$ by

$$H(x,y) := \sum_{i=1}^{n} |x_i - y_i|.$$

A fitness function $f:\{0,1\}^n \to \mathbb{R}$ can be written as a polynomial

$$f(x_1,\ldots,x_n) = \sum_{I \subseteq \{1,\ldots,n\}} c_f(I) \prod_{i \in I} x_i$$

with coefficients $c_f(I) \in \mathbb{R}$.

**Definition 4.** The *degree* of $f$ is defined as

$$\deg(f) : \max\{i \in \{0,\ldots,n\} \mid \exists I \text{ with } |I| = i \text{ and } c_f(I) \neq 0\}.$$

**Definition 5.** The *expected running time* $E(T)$ of the $(1+1)$ EA on a fitness function $f$ is defined as the mean of the number of function evaluations of Algorithm 1 before the current bit string is a global optimum of $f$. The number of function evaluations is given by the number of times line 3 is executed increased by 1 for the initial bit string. We know that the expected running time is influenced by the random choice of the initial bit string and the random mutations.

The *expected running time* of the $(1+1)$ EA on a class of fitness functions $F$ is defined as the supremum of the expected running times on $f$ for all $f \in F$.

The $(1+1)$ EA is defined without any stopping rule. Therefore, the expected running time is defined as the expected time taken to optimize the fitness function. The $(1+1)$ EA, under many different names as we mentioned above, has already been subject to various studies. As examples for theoretical investigations we mention three publications, though undoubtedly a lot of other papers can be found. Bäck [2] investigates the question of which mutation rate should be chosen. Mühlenbein [15] gives a sharp upper

bound on the expected running time on a very simple linear fitness function. A number of different upper bounds on the expected running time, also for more interesting and challenging functions, are given by Rudolph [17].

In the following section we give a general upper bound on the expected running time of the $(1+1)$ EA for all fitness functions. We demonstrate that fitness functions exist, that require that amount of time asymptotically. We present a fitness function of degree 2 that has this worst case property. In Section 3 we prove that fitness functions of degree one, i.e., linear functions, are always solvable in expected time $O(n \log n)$. In Section 4 we deal with unimodal functions which are a superclass of the linear functions discussed in Section 3. We derive lower bounds on the expected running time for two concrete unimodal fitness functions and show that this class has exponential expected running time in the worst case. After dealing with extreme running times only, in Section 5 we give a definition of $n$ functions where the expected running time for the $m$th function is $\Theta(n \log n + n^m)$ for $1 \leqslant m \leqslant n$. Finally, in Section 6 we discuss a variant of Algorithm 1 and demonstrate that very small changes in the algorithm can cause huge differences in the expected running time.

## 2. Worst case bounds and worst case examples

As we are interested in the efficiency of the $(1+1)$ EA for different fitness functions, we should compare the expected running time of the $(1+1)$ EA with other optimization algorithms. The most simple algorithm for function optimization, complete enumeration, needs $\Theta(2^n)$ steps to find the global optimum of an arbitrary fitness function over $n$ Boolean variables. In Section 2.1 we show, that the $(1+1)$ EA finds the global optimum of every fitness function on average after at most $n^n$ steps.

In the Sections 2.2 and 2.3 we present the functions DISTANCE and TRAP and prove that the expected running time of the $(1+1)$ EA for these functions equals $\Theta(n^n)$. The function DISTANCE is of degree 2 and our analysis of DISTANCE disproves the conjecture that small degree implies small running times of the $(1+1)$ EA. Nevertheless, DISTANCE is not difficult for variants of the $(1+1)$ EA. There is a probability larger than a constant $c>0$ such that the $(1+1)$ EA finds the global optimum of DISTANCE within $O(n \log n)$ steps. Hence, the parallel execution of, e.g., $n$ independent $(1+1)$ EAs will find the optimum of DISTANCE with large probability in polynomial time. If the number of parallel runs is large enough, namely $\Omega(n \log n)$, even the expected number of fitness evaluations is polynomially bounded. The function TRAP is of degree $n$ but for this function the $(1+1)$ EA needs time $\Theta(n^n)$ with a probability which is close to 1. No variant of the $(1+1)$ EA can optimize this function efficiently.

### 2.1. A general upper bound for the expected running time

**Theorem 6.** *The expected running time of the* $(1+1)$ *EA for an arbitrary fitness function is at most* $n^n$.

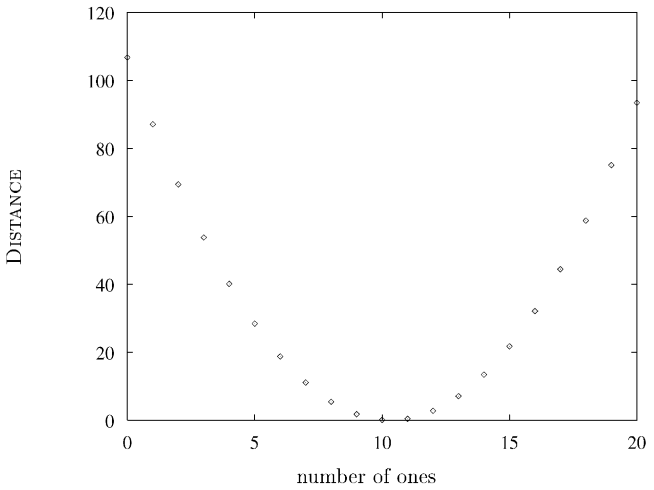


Fig. 1. The function DISTANCE for $n = 20$.

**Proof.** Let $x \in \{0,1\}^n$ be an arbitrary bit string and $x^*$ a global optimum of the fitness function. As $H(x,x^*) \leqslant n$, the probability of mutating from $x$ to $x^*$ in one step is $(1/n)^{H(x,x^*)}(1-1/n)^{n-H(x,x^*)} \geqslant n^{-n}$. Hence, the expected time until this event occurs is at most $n^n$. Because this event implies, that a global optimum of the fitness function is found, the expected running time of the $(1+1)$ EA is at most $n^n$. □

As in this proof the exact expected running time of the $(1+1)$ EA is only roughly upper bounded, it has not yet been shown that any fitness function exists such that the $(1+1)$ EA needs on average $\Theta(n^n)$ steps for optimizing it. The next subsection presents an explicit function with this property.

### 2.2. *A function of degree two with expected running time* $\Theta(n^n)$

A fitness function is assumed to be difficult to be optimized, if it gives "misleading hints" regarding the position of its global optimum. The $(1+1)$ EA most often makes only small mutations, which are accepted if they do not decrease the fitness function value. Therefore, the expected running time of the $(1+1)$ EA should be large, if for many bit strings their "neighbors", i.e., bit strings with small Hamming distance, with larger fitness lead away from the global optimum.

The function DISTANCE, defined by

$$\text{DISTANCE}(x_1,\ldots,x_n) = \left(\sum_{i=1}^{n} x_i - \left(\frac{n}{2}+\frac{1}{3}\right)\right)^2$$

is of degree 2 and has these properties (see Fig. 1 for $n = 20$).

**Theorem 7.** *The expected running time of the* $(1+1)$ *EA for* DISTANCE *equals* $\Theta(n^n)$ *but the probability that the global optimum is found within* $O(n \log n)$ *steps is bounded below by* $(1/2) - \varepsilon$ *where* $\varepsilon$ *is an arbitrary positive constant.*

**Proof.** First, we investigate the initial bit string $x$. The number of ones is binomially distributed with respect to $n$ and $1/2$. It follows by Stirling's formula that, for each $k$, the probability that $x$ contains exactly $k$ ones is bounded by $O(n^{-1/2})$. Hence, the probability that $x$ has more than $(n/2)+n^{1/4}$ ones (event $I^+$) as well as the probability that $x$ has less than $(n/2)-n^{1/4}$ ones (event $I^-$) is bounded below by $1/2-o(1)$.

It will be useful to consider also the function $x_1+\cdots+x_n$ known in the theory of EAs as OneMax (see Definition 9). It is known (Mühlenbein [15] and a simple proof is contained in Section 3) that the expected running time of the $(1+1)$ EA on OneMax is bounded above by $cn\log n$ for some constant $c$.

If the initial bit string $x$ has more than $(n/2)+n^{1/4}$ ones the $(1+1)$ EA has the same behavior on OneMax and Distance as long as no bit string $x'$ with at most $(n/2)-n^{1/4}$ ones is created by mutation.

Starting with more than $(n/2)+n^{1/4}$ ones, the $(1+1)$ EA accepts for Distance as well as for OneMax no bit string where the number of ones is larger than $(n/2)-n^{1/4}$ and at most $(n/2)+n^{1/4}$. Hence, a bit string $x'$ with at most $(n/2)-n^{1/4}$ ones is only created if at least $2n^{1/4}$ bits flip simultaneously. The number of flipping bits is binomially distributed with respect to $n$ and $1/n$ and this distribution can be approximated by the Poisson distribution with parameter $\lambda=1$. Hence, the probability of at least $2n^{1/4}$ flipping bits is bounded above by $2^{-\Omega(n^{1/4}\log n)}$. Let $E$ be the event that during the first $n^2$ steps it never happens that at least $2n^{1/4}$ bits flip simultaneously. The probability for this event equals $1-n^2\cdot 2^{-\Omega(n^{1/4}\log n)}=1-2^{-\Omega(n^{1/4}\log n)}$.

The upper bound $cn\log n$ on the expected running time of the $(1+1)$ EA on OneMax also holds under the condition $I^+\cap E$ which has a probability of $1/2-o(1)$. This follows from the simple investigation of OneMax in Section 3. By Markoff's inequality, the $(1+1)$ EA reaches the optimal bit string for OneMax $x_{\text{one}}$ consisting of ones only under the condition $I^+\cap E$ with probability at least $1-1/c'$ within $c'cn\log n$ steps. Under the same condition $I^+\cap E$, the $(1+1)$ EA on Distance reaches $x_{\text{one}}$ with probability at least $1-1/c'$ within $c'cn\log n$ steps. Hence, with probability at least $1/2-1/c'-o(1)$ the $(1+1)$ EA reaches $x_{\text{one}}$ within $c'cn\log n$ and then needs on average $n^n$ steps to reach the optimal bit string $x_{\text{zero}}$ consisting of zeros only. This implies the result on the expected running time.

The same arguments work if we replace $I^+$ by $I^-$. Hence, with probability at least $1/2-1/c'-o(1)$ the $(1+1)$ EA reaches the optimal string $x_{\text{zero}}$ within $c'cn\log n$ steps. The second claim follows if we choose $c'$ large enough. □

### 2.3. A function where the running time is $\Theta(n^n)$ with large probability

The function Trap: $\{0,1\}^n\to\mathbb{R}$ defined by

$$\mathrm{TRAP}(x_1,\ldots,x_n):=\sum_{i=1}^{n}x_i+(n+1)\cdot\prod_{i=1}^{n}(1-x_i)$$

has been introduced in [1]. Its degree is $n$ and $\mathrm{Trap}(x)=\mathrm{OneMax}(x)$, if $x\neq x_{\text{zero}}$. The global optimum of Trap is $x_{\text{zero}}$.

**Theorem 8.** *For each constant $c<1$ there exists a constant $\varepsilon>0$ such that the $(1+1)$ EA needs for* TRAP *with probability at least $c$ at least $\varepsilon n^n$ steps.*

**Proof.** The probability that the initial bit string $x$ has at least $n/4$ ones is $1-o(1)$ (Chernoff's inequality, see Motwani and Raghavan [14]). The probability that within $n^2$ steps there is no step where at least $n/4$ bits flip simultaneously is $1-o(1)$ (see the proof of Theorem 7). Since TRAP and ONEMAX only differ on $x_{\text{zero}}$, we conclude that with probability $1-o(1)$ the $(1+1)$ EA reaches $x_{\text{one}}$ within $n^2$ steps. Here we have applied Markoff's inequality on the expected running time of the $(1+1)$ EA for ONEMAX. Having reached $x_{\text{one}}$ only $x_{\text{zero}}$ is accepted as a different bit string. The probability that $x_{\text{zero}}$ is not created during $\varepsilon n^n-1$ mutations of $x_{\text{one}}$ equals $(1-n^{-n})^{\varepsilon n^n-1} \geqslant e^{-\varepsilon}$. We choose $\varepsilon$ small enough such that the error probabilities $o(1)$ and $1-(1-n^{-n})^{\varepsilon n^n-1}$ are less than $1-c$. □

## 3. Linear functions[1]

We know from the previous section that fitness functions of degree at least 2 can be very difficult for the $(1+1)$ EA. Fitness functions of degree 0 are, of course, trivial, since they are constant. Fitness functions of degree 1 remain as an interesting class of functions, and they are in fact easy for the $(1+1)$ EA as we will prove here. We call these functions *linear*, since they can be written as

$$f(x) = \tau + \sum_{i=1}^{n} w_i x_i \tag{1}$$

for some $\tau, w_i \in \mathbb{R}$.

For the analysis in this section we make without loss of generality some assumptions. Constant additive terms have no influence on the behavior of the $(1+1)$ EA, so we assume $\tau = 0$. We assume that all weights are non-negative. Otherwise one may replace $x_i$ by $1-x_i$. Furthermore, the weights $w_i$ are assumed to be integers. Finally, we assume that the weights are sorted, i.e., $w_1 \geqslant \cdots \geqslant w_n$. It follows that $x_{\text{one}}$ is always a global optimum. Moreover, if all weights are positive, $x_{\text{one}}$ is the only global optimum.

Before we consider an arbitrary linear function $f$, we introduce two special linear functions that are of particular interest.

**Definition 9.** The linear function ONEMAX has all weights set to 1, i.e.,

$$\text{ONEMAX}(x) = \sum_{i=1}^{n} x_i.$$

The linear function BIN has set the $i$th weight according to $w_i := 2^{n-i}$, i.e., BIN interprets a bit string as binary representation of an integer.

[1] The results of this section have been presented at the ICEC '98 [3].

These two functions are in some sense two extreme examples from the class of linear functions. The weights of BIN are so strongly decreasing that $w_i > \sum_{j=i+1}^{n} w_j$ holds for all $i$ with $1 \leqslant i < n$. This implies that it is always the leftmost flipping bit alone that decides whether a mutation is accepted. The weights of ONEMAX are all equal, so that it is only the number of bits flipping to one compared with the number of bits flipping to zero that decides whether a mutation is accepted.

The function ONEMAX is easy to analyze; upper bounds for the expected running time have been presented, e.g., by Mühlenbein [15]. As for all symmetric functions, mutation steps to bit strings with equal number of ones can be ignored. In successful steps the number of ones is increased by at least 1, so at most $n$ successful steps are sufficient. If the number of zeros in the current bit string equals $i$, the probability for a successful step is bounded below by $\binom{i}{1} n^{-1}(1-1/n)^{n-1}$, so we get

$$\sum_{i=1}^{n} \left( \binom{i}{1} \frac{1}{n} \left(1 - \frac{1}{n}\right)^{n-1} \right)^{-1} \leqslant en \sum_{i=1}^{n} \frac{1}{i} = \mathrm{O}(n \ln n)$$

as an upper bound for the expected running time. A lower bound of equal order of growth is easy to find, too. Furthermore, it is valid for all linear functions with all non-zero weights.

**Lemma 10.** *The expected number of steps the* $(1+1)$ *EA takes to optimize a linear function with all non-zero weights is* $\Omega(n \ln n)$.

**Proof.** By our assumptions all weights $w_i$ are positive. Since $x_{\text{one}}$ is the only optimum, it is necessary that each bit that is zero after random initialization flips at least once. Hence, the average time until each of these bits has tried to flip at least once is a lower bound on the considered expected time. The following considerations are similar to the example called "coupons collector's problem" by Motwani and Raghavan [14].

Let $T$ denote the random variable describing the first point of time where each of these bits has tried to flip at least once. Since $T$ takes only positive integers, we have

$$E(T) = \sum_{t=1}^{\infty} t \,\mathrm{Prob}(T = t) = \sum_{t=1}^{\infty} \mathrm{Prob}(T \geqslant t).$$

Without loss of generality $n$ is even. With probability at least $1/2$ at least half of the bits are zero after random initialization. $(1-1/n)^{t-1}$ describes the probability that one bit does not flip at all in $t-1$ steps. So, $1-(1-1/n)^{t-1}$ is the probability that it flips at least once in $t-1$ steps. Therefore, we have $(1-(1-1/n)^{t-1})^{n/2}$ as probability that this is the case with $n/2$ bits. Finally, $1-(1-(1-1/n)^{t-1})^{n/2}$ is the probability for the event that at least one of $n/2$ bits never flips in $t-1$ steps. So, we have

$$E(T) \geqslant \frac{1}{2} \sum_{t=1}^{\infty} \left(1 - \left(1 - \left(1 - \frac{1}{n}\right)^{t-1}\right)^{n/2}\right)$$

$$\geqslant \frac{1}{2}(n-1)(\ln n)\left(1-\left(1-\left(1-\frac{1}{n}\right)^{(n-1)\ln n}\right)^{n/2}\right)$$

$$\geqslant \frac{1}{2}(n-1)(\ln n)(1-\mathrm{e}^{-1/2}) = \Omega(n\ln n). \qquad \square$$

For BIN an upper bound is harder to find. Since it is only the leftmost flipping bit that decides whether a mutation is accepted, the Hamming distance to $x_{\text{one}}$ may be increased during optimization. Even extreme steps like the one from $(0,1,1,\ldots,1)$ to $(1,0,0,\ldots,0)$ are possible though extremely unlikely.

**Lemma 11.** *The expected number of steps of the* $(1+1)$ *EA on the function* BIN *is* $\Theta(n \ln n)$.

**Proof.** The lower bound is given in Lemma 10, so we only prove an upper bound.

Without loss of generality $n$ is even. Let $T$ be the random variable describing the first point of time where the $(1+1)$ EA reaches $x_{\text{one}}$. We cut up the optimization process and distinguish the first phase of length $T_1$ until the leftmost $n/2$ bits all are ones and the second phase of length $T_2 = T - T_1$. The reasons for this will be discussed later.

In order to derive an upper bound on $E(T_1)$ we distinguish the mutations: steps that change at least one of the bits in the left half are called *successful*. Unsuccessful steps may only change bits in the right half and are not important for us.

Let $X_i$ be the random variable describing the first point of time where the left half of the current bit string $x$ contains at least $i$ ones, in particular $X_0 = 0$. Then we have

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

We distinguish between successful and unsuccessful steps and therefore introduce the random variable $Y_i$ that describes the random number of successful steps in the interval $[X_{i-1}+1,\ldots,X_i]$, as well as $Z_i$ that describes the random number of unsuccessful steps during that period of time. This yields

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

and we can investigate successful and unsuccessful steps separately.

We claim that $E(Y_i) \leqslant 2$. The right half of the bit string has no influence on the random variable $Y_i$ and we investigate only the bit strings of the $n/2$ leftmost bits. We start with a bit string $x_0$ with $i-1$ ones and have to estimate the number of successful steps until we obtain for the first time a bit string with more than $i-1$ ones. Hence, we only investigate the $(1+1)$ EA on bit strings with at most $i-1$ ones.

Let $x_0, x_1, x_2, \ldots$ be the sequence of different bit strings produced by the $(1+1)$ EA starting on $x_0$. Let $D_j$ be the random variable describing the difference of the number of ones of $x_j$ and $x_{j-1}$. Then $Y_i$ is the first point of time $t$ where $D_1 + \cdots + D_t \geqslant 1$. The

probability distribution of $D_j$ under the assumption $D_1+\cdots+D_{j-1}<1$ can be described as follows. The bit string $x_{j-1}$ has at most $i-1$ ones. The step is successful if and only if the leftmost flipping bit is a bit flipping from 0 to 1. This position $p$ contributes 1 to $D_j$. Let $k \leqslant n/2-1$ be the number of ones of $x_{j-1}$ to the right of position $p$ and $l$ the corresponding number of zeros. Under all the considered assumptions $D_j = 1 - B_j$ where $B_j = B_{j,0} - B_{j,1}$ and $B_{j,0}$ is binomially distributed with respect to $k$ and $1/n$ and $B_{j,1}$ is binomially distributed with respect to $l$ and $1/n$.

We introduce random variables which are easier to handle and which can be compared with $D_j$. Let $B_j^*$ be independent random variables which are binomially distributed with respect to $n/2-1$ and $1/n$ and let $D_j^* = 1 - B_j^*$.

It follows that for all possible values of $n, j, p, k, l$, and $r$ and independently of the values of $B_1,\ldots,B_{j-1}$ $\mathrm{Prob}(B_j^* \leqslant r) \leqslant \mathrm{Prob}(B_j \leqslant r)$ and, therefore, $\mathrm{Prob}(D_j^* \leqslant r) \geqslant \mathrm{Prob}(D_j \leqslant r)$. Let $T^*$ be the first point of time $t$ where $D_1^* + \cdots + D_t^* \geqslant 1$. Then $\mathrm{Prob}(T^* \leqslant t^*) \leqslant \mathrm{Prob}(Y_i \leqslant t^*)$ for all $t^*$ and $E(T^*) \geqslant E(Y_i)$. Hence, it is sufficient to prove that $E(T^*) \leqslant 2$.

By definition, $D_j^* \leqslant 1$ and $E(D_j^*) = 1 - (n/2-1)/n > 1/2$. Let $S_t = D_1^* + \cdots + D_t^*$. The random variables $S_0 = 0, S_1, S_2, \ldots$ describe a random walk on the line which is homogeneous with respect to time and place. Since $D_j^* \leqslant 1$, we reach 1 as the first point to the right of the point 0. We stop the random walk whenever we reach the point 1 and $T^*$ is the stopping time of this process. After the first step of the random walk the distance to point 1 is $1 - D_1^*$ and, because of the homogeneity of the process, the expected stopping time starting at distance $d$ from point 1 equals $\mathrm{d}E(T^*)$. Hence,

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

We conclude that $E(T^*) = 1/E(D_1^*)$, and with $E(D_k^*) > 1/2$ for all $k$ we have $E(T^*) \leqslant 2$. This implies $E(Y_i) \leqslant 2$ for all $i$. The equality $E(T^*)E(D_1^*) = 1$ is known in more general form in probability theory as Wald's identity [4]. We remark that if we start the same proof method without restricting ourselves to the left half of the bit strings, $B_j^*$ is binomially distributed for $n-1$ and $1/n$ and $E(D_j^*) = 1/n$ is too small to obtain the desired result.

Now, we look for some lower bound $p$, for the probability of a successful mutation. Given $p$, the average time until we have $k$ successful mutations is at most $k/p$, so we have

$$\begin{aligned} E(Y_i + Z_i) &= \sum_k \mathrm{Prob}(Y_i = k)E(Y_i + Z_i \mid Y_i = k) \\ &= \sum_k \mathrm{Prob}(Y_i = k)\frac{k}{p} = \frac{E(Y_i)}{p} \leqslant \frac{2}{p}. \end{aligned}$$

During the steps $X_{i-1}+1,\ldots,X_i$ the number of ones in the left half of $x$ is bounded above by $i-1$. A sufficient condition for a successful step is that all bits equal to one do not try to flip and exactly one of the bits equal to zero tries to flip. The probability of this event is bounded below by

$$\binom{n/2-(i-1)}{1}\frac{1}{n}\left(1-\frac{1}{n}\right)^{n/2-1} \geqslant \left(\frac{n}{2}-i+1\right)\frac{\mathrm{e}^{-1/2}}{n}.$$

Altogether we have

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

$$= 2\mathrm{e}^{1/2}n\sum_{i=1}^{n/2}\frac{1}{i} \leqslant 2\mathrm{e}^{1/2}n(\ln n+1) = \mathrm{O}(n\ln n).$$

The investigation of the second phase can be carried out analogously. Only the probability of a successful step changes, since it is necessary to guarantee that no bit of the left half of $x$ tries to flip. We achieve this by replacing $(1-1/n)^{n/2-1}$ by $(1-1/n)^{n-1}$. This yields the upper bound $2en(\ln n+1)$, so we have

$$E(T) = E(T_1)+E(T_2) \leqslant 2(\mathrm{e}+\mathrm{e}^{1/2})n(\ln n+1) = \mathrm{O}(n\ln n). \qquad \square$$

For an arbitrary linear function things are a little more complicated. On the one hand, a bit that flips from zero to one can have such a large weight that it allows several other bits to flip from 1 to 0 simultaneously, as it is the case for BIN. On the other hand, different to BIN, it may well be true that leading ones are not guaranteed to remain unchanged, as it is the case with ONEMAX. But the main idea from the proof of the upper bound on the expected running time for BIN can be carried over: one can distinguish more and less important bits according to their weights.

**Theorem 12.** *The expected running time of the* $(1+1)$ *EA on the class of linear functions with non-zero weights is* $\Theta(n\ln n)$.

**Proof.** The lower bound follows from Lemma 10. We prove the upper bound for an arbitrary linear function $f$. We have already discussed that without loss of generality $\tau=0$ and $w_1 \geqslant w_2 \geqslant \cdots \geqslant w_n > 0$. We follow the main ideas of the proof of Lemma 11 but the general situation is much more complicated.

We measure the progress of the $(1+1)$ EA on the function $f$ by the artificial fitness function

$$\mathrm{val}(x) = 2\sum_{i=1}^{n/2} x_i + \sum_{i=n/2+1}^{n} x_i.$$

This artificial fitness function plays the role of a potential function $\Phi$ in the analysis of data structures and algorithms. We investigate the $(1+1)$ EA on $f$ but we measure

the progress with respect to val. The aim is to prove that steps where the actual string changes lead to an expected gain which is bounded below by a positive constant. Therefore, it is necessary that the potential function contains the information that the first half positions are those with the larger weights.

The initial bit string $x$ has the value $\mathrm{val}(x)$ which is not negative. The value of the global optimum equals $(3/2)n$. A step of the $(1+1)$ EA working on the linear function $f$ is called *successful* if it produces from $x$ a mutated string $x'$ which replaces $x$ (i.e., $x' \neq x$ and $f(x') \geqslant f(x)$). Let $x$ be a bit string with $\mathrm{val}(x) = i < (3/2)n$. We are interested in the expected number of successful steps until we obtain for the first time a bit string $x'$ where $\mathrm{val}(x') > \mathrm{val}(x)$. Later we prove that we can upper bound the expected number independently from $x$ by a constant $c^*$. Afterwards, it is easy to bound the expected number of successful and unsuccessful steps. This can be done in the same way as in the proof of Lemma 11 by proving a lower bound on the probability of successful steps if the value of the bit string is bounded above by $i < (3/2)n$. Then the number of zeros in the bit string is at least $r_i = \lceil (3/4)n - (1/2)i \rceil$ and there exist at least $r_i$ different one bit mutations which increase $f$. Hence, the success probability is at least $r_i(1/n)(1-1/n)^{n-1} \geqslant \mathrm{e}^{-1} r_i/n$. Altogether, the expected number of steps to optimize $f$ is bounded by

$$\sum_{i=0}^{(3/2)n-1} c^* \frac{1}{r_i} \mathrm{e}n \leqslant 2\,c^* \mathrm{e}n2 \cdot \sum_{j=1}^{\lceil (3/4)n \rceil} \frac{1}{j} = \mathrm{O}(n \ln n).$$

The only claim we have to prove is the existence of a constant upper bound on the expectation of the number of successful steps to increase the value. Let $x$ be a bit string (not the global optimum) and $x'$ the random bit string produced by a successful step based on $x$. Let $S$ be the random set of indices $i$ where $x_i = 0$ and $x'_i = 1$. Since the step was successful, $S \neq \emptyset$. Let $D_S(x) = \mathrm{val}(x') - \mathrm{val}(x)$ be the random variable describing for fixed $S$ the gain with respect to the value function. Our main step is to define a random variable $D^*_S(x)$ which takes integer values, $D^*_S(x) \leqslant 1$, and has the property that $\mathrm{Prob}(D^*_S(x) \leqslant r) \geqslant \mathrm{Prob}(D_S(x) \leqslant r)$ for all $r$. We prove that there is some positive constant $d^*$ (which does neither depend on $x$ nor on $S$), such that $E(D^*_S(x)) \geqslant d^*$ holds. In order to do so, we consider finitely many cases and choose at the end the smallest of the considered constants.

The definition of $D^*_S(x)$ is complicated. Let us consider $D_S(x) = \mathrm{val}(x') - \mathrm{val}(x)$. The contribution of positions $i$ to $D_S(x)$ equals

- $+2$, if $x_i = 0$, $x'_i = 1$, and $i \leqslant n/2$,
- $-2$, if $x_i = 1$, $x'_i = 0$, and $i \leqslant n/2$,
- $+1$, if $x_i = 0$, $x'_i = 1$, and $i > n/2$,
- $-1$, if $x_i = 1$, $x'_i = 0$, and $i > n/2$,
- 0, if $x_i = x'_i$.

Let $p$ be the minimal element of $S$. If $p \leqslant n/2$, we only take the negative contributions of mutations into account and define

$$d^*_S(x) := 2 - 2|\{i \leqslant n/2 \mid x_i = 1, x'_i = 0\}| - |\{i > n/2 \mid x_i = 1, x'_i = 0\}|.$$

Then $D_S^*(x) := \min\{1, d_S^*(x)\}$. If $p > n/2$, we have to be more careful, because every bit flipping from zero to one has only a contribution of 1. Hence, we define $D_S^*(x) := \min\{1, \mathrm{val}(x') - \mathrm{val}(x)\}$.

Is is obvious that, in both cases, $D_S^*(x)$ is an integer, $D_S^*(x) \leqslant 1$, and $D_S^*(x) \leqslant D_S(x)$ implying that $\mathrm{Prob}(D_S^*(x) \leqslant r) \geqslant \mathrm{Prob}(D_S(x) \leqslant r)$. We have to prove that $E(D_S^*(x)) \geqslant d^*$ for some positive constant $d^*$.

*Case* 1. $p \leqslant n/2$: We know that the mutation from $x$ to $x'$ is a successful one and that $S$ is the set of all indices $i$ where $x_i = 0$ and $x_i' = 1$. Hence, $x_i = 0$ and $i \notin S$ implies $x_i' = 0$.

Let $j_2$ be the number of positions $i \leqslant n/2$ where $x_i = 1$ and the mutation where only $x_i$ flips from 1 to 0 is successful, and let $j_1$ be the corresponding number of positions $i > n/2$. The expected number of flipping bits among these $j_2 + j_1$ bits (under no condition) is $(j_2 + j_1)/n$ which leads to an expected contribution of $-(2j_2 + j_1)/n$ to $d_S^*(x)$. Given the set $S$ of bits flipping from 0 to 1, if we assume in addition that the mutation is successful, this may only decrease the number of bits flipping from 1 to 0. Therefore, we may estimate the expected contribution of these bits by $-(2j_2 + j_1)/n$. We have shown that $E(d_S^*(x)) \geqslant 2 - (2j_2 + j_1)/n$. If no bit flips from 1 to 0, $d_S^*(x) = 2$ but $D_S^*(x) = 1$. We have to consider the conditional probability that no bit flips from 1 to 0. It is sufficient to consider the $j_2 + j_1$ selected positions. Let $C$ be the event that the mutation is successful and $A$ be the event that no bit flips from 1 to 0. Since $A$ implies $C$ we have $\mathrm{Prob}(A|C) = \mathrm{Prob}(A)/\mathrm{Prob}(C)$. Obviously, $\mathrm{Prob}(A) = (1 - 1/n)^{j_2 + j_1}$. We are interested in upper bounds on $\mathrm{Prob}(A|C)$ and, therefore, in lower bounds on $\mathrm{Prob}(C)$. By definition

$$\mathrm{Prob}(C) \geqslant \left(1 - \frac{1}{n}\right)^{j_2 + j_1} + (j_2 + j_1)\frac{1}{n}\left(1 - \frac{1}{n}\right)^{j_2 + j_1 - 1}$$

implying that

$$\mathrm{Prob}(A|C) \leqslant \frac{1 - 1/n}{1 - 1/n + (j_2 + j_1)/n} \leqslant \frac{1}{1 + (j_2 + j_1)/n}.$$

We conclude that

$$E(D_S^*(x)) \geqslant E(d_S^*(x)) - \mathrm{Prob}(A|C) \geqslant 2 - \frac{2j_2 + j_1}{n} - \frac{1}{1 + (j_2 + j_1)/n}.$$

This lower bound for $E(D_S^*(x))$ is a decreasing function of $j_2$ and $j_1$. Since $j_2 \leqslant (n/2) - 1$ by assumption, we can consider the lower bound for $j_2 = n/2$ leading to

$$E(D_S^*(x)) \geqslant 1 - \frac{j_1}{n} - \frac{1}{(3/2) + (j_1/n)}.$$

If $j_1 \leqslant n/3$, $E(D_S^*(x)) \geqslant 4/33$. If $j_1 > n/3$, we consider the $v = \binom{j_2 + j_1}{2}$ pairs of the considered positions and denote by $v'$ the number of these pairs which may flip from 1 to 0 without making the mutation unsuccessful. If $v' \geqslant v/2$, the lower bound on $\mathrm{Prob}(C)$ can be increased by $(v/2)(1/n^2)(1 - 1/n)^{j_2 + j_1 - 2}$ which is larger than some positive constant.

Since $j_1/n \leqslant 1/2$, $\mathrm{Prob}(A|C) \leqslant (1/2) - \delta$ for some positive constant $\delta$, and we are done. If $v' < v/2$, we again use the estimation $\mathrm{Prob}(A|C) \leqslant 1/2$. In this case the upper bound on the expected number of bits flipping from 0 to 1 can be improved. The number of pairs which are not allowed to flip is at least $\alpha n^2$ for some constant $\alpha > 0$ and the probability that exactly one of these pairs tries to flip from 1 to 0 is bounded below by some constant $\alpha' > 0$. Hence, the expected contribution of the bits flipping from 1 to 0 to $D_S^*(x)$ can be lower bounded by $-(2j_2 + j_1)/n + \alpha'' \geqslant -3/2 + \alpha$" for some constant $\alpha'' > 0$. Altogether, we have proved that $E(D_S^*(x)) \geqslant d^*$ for some positive constant $d^*$ in Case 1.

*Case* 2. $p > n/2$: Let $s = |S|$. First, we consider the subcase $s \geqslant 4$. In order to prove a lower bound on $E(D_S^*(x))$, we work with the following assumptions which only lead to smaller values of $E(D_S^*(x))$. We do not make use of the fact that the step from $x$ to $x'$ has to be successful and we assume that $x_i = 1$ for all $i \notin S$. Then we have $n/2$ positions $i \leqslant n/2$ where $x_i = 1$ and less then $n/2$ positions $i > n/2$ where $x_i = 1$. We overestimate the number of bits flipping from 1 to 0 by assuming that we have $n/2$ positions $i > n/2$ where $x_i = 1$. The random number $Z_2'$ of bits in the first half flipping from 1 to 0 is binomially distributed with respect to the parameters $n/2$ and $1/n$. The same holds by our pessimistic assumptions for the random number $Z_1'$ of bits in the second half. It is sufficient to prove a lower bound on $E(D_S^*(x))$ under the assumption that the number of bits flipping from 1 to 0 are given by $Z_2'$ and $Z_1'$, respectively. The binomial distribution with parameters $n/2$ and $1/n$ can be approximated by the Poisson distribution with parameter $\lambda = 1/2$. Let $Z_2$ and $Z_1$ be independent random variables whose distribution is Poisson with parameter $\lambda = 1/2$. We work with $Z_2$ and $Z_1$ instead of $Z_2'$ and $Z_1'$, respectively. Later we discuss the mistake caused by this replacement.

We know that

$$\mathrm{Prob}(Z_2 = z_2, Z_1 = z_1) = \frac{1}{z_2!}\left(\frac{1}{2}\right)^{z_2} \mathrm{e}^{-1/2} \frac{1}{z_1!}\left(\frac{1}{2}\right)^{z_1} \mathrm{e}^{-1/2}.$$

In the following we estimate $E(D_S^*(x))$ under the assumption that the numbers of bits flipping from 1 to 0 are given by $Z_2$ and $Z_1$, respectively. This new random variable is called $D_S^{**}(x)$. The pair $(Z_2, Z_1)$ of random variables takes values from $\mathbb{N}_0 \times \mathbb{N}_0$. Hence,

$$E(D_S^{**}(x)) = \sum_{z_2 \geqslant 0, z_1 \geqslant 0} E(D_S^{**}(x) \,|\, Z_2 = z_2, Z_1 = z_1)\, \mathrm{Prob}(Z_2 = z_2, Z_1 = z_1). \quad (*)$$

We partition $\mathbb{N}_0 \times \mathbb{N}_0$ into disjoint sets:

- $A_1 = \{(0,0)\}$,
- $A_2 = \{(0, z_1) \,|\, z_1 \geqslant 5\}$,
- $A_3 = \{(1, z_1) \,|\, z_1 \geqslant 3\}$,
- $A_4 = \{(z_2, z_1) \,|\, z_2 \geqslant 2, z_1 \geqslant 0\}$,
- $A_5 = \mathbb{N}_0 \times \mathbb{N}_0 \backslash (A_1 \cup A_2 \cup A_3 \cup A_4)$.

It is easy to see that

- $D_S^{**}(x)=1$, if $(z_2,z_1)\in A_1$,
- $D_S^{**}(x)\geqslant 4-z_1$, if $(z_2,z_1)\in A_2$,
- $D_S^{**}(x)\geqslant 2-z_1$, if $(z_2,z_1)\in A_3$,
- $D_S^{**}(x)\geqslant 4-2z_2-z_1$, if $(z_2,z_1)\in A_4$,
- $D_S^{**}(x)\geqslant 0$, if $(z_2,z_1)\in A_5$.

Let $\mathrm{con}_i$, $1\leqslant i\leqslant 5$, be the contribution of all terms $(z_2,z_1)\in A_i$ to the sum $(*)$. Then

- $\mathrm{con}_1\geqslant \mathrm{e}^{-1}$,
- $\mathrm{con}_2\geqslant \sum_{z_1\geqslant 5}(4-z_1)(1/z_1!)(\frac{1}{2})^{z_1}\mathrm{e}^{-1}\geqslant -\frac{1}{1920}\mathrm{e}^{-1}$,
- $\mathrm{con}_3\geqslant \sum_{z_1\geqslant 3}(2-z_1)(1/z_1!)(\frac{1}{2})^{z_1+1}\mathrm{e}^{-1}\geqslant -\frac{1}{48}\mathrm{e}^{-1}$,
- $\mathrm{con}_4\geqslant \sum_{z_2\geqslant 2}\sum_{z_1\geqslant 0}(4-2z_2-z_1)(1/z_2!z_1!)(\frac{1}{2})^{z_2+z_1}\mathrm{e}^{-1}\geqslant -\frac{53}{192}\mathrm{e}^{-1}$,
- $\mathrm{con}_5\geqslant 0$.

Hence,

$$E(D_S^{**}(x))\geqslant \frac{1349}{1920}\mathrm{e}^{-1}.$$

However, we are interested in an estimate of $E(D_S^{*}(x))$. We estimate the mistake by replacing the random variables $Z_1'$ and $Z_2'$ which are binomially distributed with respect to $n/2$ and $1/n$ with the random variables $Z_1$ and $Z_2$, respectively, with a Poisson distribution with respect to $1/2$. In the subcase $s\geqslant 4$ as in the later subcases we only consider terms

$$\mathrm{Prob}(Z_i'=r) \text{ and } r\,\mathrm{Prob}(Z_i'=r) \quad \text{for some } r\leqslant 4,$$

$$\sum_{j\geqslant r}\mathrm{Prob}(Z_i'=j) \text{ and } \sum_{j\geqslant r} j\,\mathrm{Prob}(Z_i'=j) \text{ for some } r\leqslant 5$$

and finite combinations of these terms. It is well-known that for constant $\lambda$ and parameters $n$ and $p(n)$ for the binomial distribution where $np(n)\to\lambda$ as $n\to\infty$ the binomial distribution converges against the Poisson distribution. In our case the product of the parameters of the binomial distribution equals $\lambda=1/2$. Hence, for each $\varepsilon>0$ and $n$ large enough

$$|\mathrm{Prob}(Z_i'=r)-\mathrm{Prob}(Z_i=r)|\leqslant\varepsilon$$

and

$$|r\,\mathrm{Prob}(Z_i'=r)-r\,\mathrm{Prob}(Z_i=r)|\leqslant\varepsilon$$

for all $r\leqslant 4$. Since

$$\sum_{j\geqslant 0}\mathrm{Prob}(Z_i'=j)=\sum_{j\geqslant 0}\mathrm{Prob}(Z_i=j)=1$$

and

$$\sum_{j\geqslant 0} j\,\mathrm{Prob}(Z_i'=j)=\sum_{j\geqslant 0} j\,\mathrm{Prob}(Z_i=j)=1/2,$$

we can also estimate the corresponding differences of the sums $\sum_{j\geqslant r}$ by $\varepsilon$. Hence, for large $n$, the mistake in each $\mathrm{con}_i$ is smaller than any given constant and this holds also for the sum of a constant number of these mistakes.

For the subcases $s \in \{1,2,3\}$, we are more careful by excluding cases which necessarily lead to unsuccessful steps. A step cannot be successful if $z_2 > s$ or $z_2 = s$ and $z_1 > 0$. Hence, we consider only steps where $z_2 \leqslant s-1$ or $z_2 = s$ and $z_1 = 0$. Let $\alpha_s$ be the probability that $z_2 \leqslant s-1$ or $z_2 = s$ and $z_1 = 0$. We follow the same approach as in the subcase $s \geqslant 4$, the probability of the event $(Z_2 = z_2, Z_1 = z_1)$ under the condition $Z_2 \leqslant s-1$ or $Z_2 = s$ and $Z_1 = 0$ is equal to the product of $\alpha_s^{-1}$ and the unconditional probability of the event $(Z_2 = z_2, Z_1 = z_1)$.

For $s = 1$, we partition the set of pairs $(z_2, z_1)$ where $z_2 \leqslant 0$ or $z_2 = 1$ and $z_1 = 0$ into
- $A_1 = \{(0,0)\}$,
- $A_2 = \{(1,0)\}$,
- $A_3 = \{(0,z_1) \mid z_1 \geqslant 1\}$.

It is easy to see that
- $D_S^{**}(x) = 1$, if $(z_2, z_1) \in A_1$,
- $D_S^{**}(x) = -1$, if $(z_2, z_1) \in A_2$,
- $D_S^{**}(x) = 1 - z_1$, if $(z_2, z_1) \in A_3$.

Then
- $\mathrm{con}_1 = \mathrm{e}^{-1}\alpha_1^{-1}$,
- $\mathrm{con}_2 = -\frac{1}{2}\mathrm{e}^{-1}\alpha_1^{-1}$,
- $\mathrm{con}_3 \geqslant \sum_{z_1 \geqslant 1}(1-z_1)(1/z_1!)(\frac{1}{2})^{z_1}\mathrm{e}^{-1}\alpha_1^{-1} \geqslant -\frac{1}{4}\mathrm{e}^{-1}\alpha_1^{-1}$.

Hence, $E(D_S^*(x))$ is also in this subcase bounded below by a positive constant, if $n$ is large enough.

For $s = 2$ we partition the set of pairs $(z_2, z_1)$ where $z_2 \leqslant 1$ or $z_2 = 2$ and $z_1 = 0$ into
- $A_1 = \{(0,0)\}$,
- $A_2 = \{(1,0),(2,0)\}$,
- $A_3 = \{(0,z_1),(1,z_1) \mid z_1 \geqslant 1\}$.

It is not difficult to see that we have
- $D_S^{**}(x) = 1$, if $(z_2, z_1) \in A_1$,
- $D_S^{**}(x) = 2 - 2z_2$, if $(z_2, z_1) \in A_2$,
- $D_S^{**}(x) = 2 - 2z_2 - z_1$, if $(z_2, z_1) \in A_3$.

Then
- $\mathrm{con}_1 = \mathrm{e}^{-1}\alpha_2^{-1}$,
- $\mathrm{con}_2 = -2\frac{1}{8}\mathrm{e}^{-1}\alpha_2^{-1} = -\frac{1}{4}\mathrm{e}^{-1}\alpha_2^{-1}$,
-

$$\mathrm{con}_3 \geqslant \left(\sum_{z_1 \geqslant 2}(2-z_1)\frac{1}{z_1!}\left(\frac{1}{2}\right)^{z_1}\mathrm{e}^{-1}\alpha_2^{-1}\right) + \left(\sum_{z_1 \geqslant 1} -z_1\frac{1}{(z_1)!}\left(\frac{1}{2}\right)^{z_1+1}\mathrm{e}^{-1}\alpha_2^{-1}\right)$$

$$\geqslant \left(-\frac{1}{24} - \frac{5}{12}\right)\mathrm{e}^{-1}\alpha_2^{-1} = -\frac{11}{24}\mathrm{e}^{-1}\alpha_2^{-1}.$$

Thereby, $E(D_S^*(x))$ is lower bounded by a positive constant in this subcase, too.

For $s = 3$ we partition the set of pairs $(z_2, z_1)$ where $z_1 \leqslant 2$ or $z_2 = 3$ and $z_1 = 0$ into
- $A_1 = \{(0,0)\}$,
- $A_2 = \{(0,z_1) \mid z_1 \geqslant 3\}$,

- $A_3 = \{(1,z_1) \mid z_1 \geqslant 1\}$,
- $A_4 = \{(2,z_1) \mid z_1 \geqslant 0\}$,
- $A_5 = \{(3,0)\}$,
- $A_6 = \{(1,0),(0,1),(0,2)\}$.

It is easy to see that

- $D_S^{**}(x) = 1$, if $(z_2,z_1) \in A_1$,
- $D_S^{**}(x) = 3 - z_1$, if $(z_2,z_1) \in A_2$,
- $D_S^{**}(x) = 1 - z_1$, if $(z_2,z_1) \in A_3$,
- $D_S^{**}(x) = -1 - z_1$, if $(z_2,z_1) \in A_4$,
- $D_S^{**}(x) = -3$, if $(z_2,z_1) \in A_5$,
- $D_S^{**}(x) \geqslant 0$, if $(z_1,z_2) \in A_6$.

This leads to

- $\mathrm{con}_1 = \mathrm{e}^{-1}\alpha_3^{-1}$,
- $\mathrm{con}_2 = \sum_{z_1 \geqslant 3}(3 - z_1)(1/z_1!)(\frac{1}{2})^{z_1}\mathrm{e}^{-1}\alpha_3^{-1} \geqslant -\frac{1}{192}\mathrm{e}^{-1}\alpha_3^{-1}$,
- $\mathrm{con}_3 = \sum_{z_1 \geqslant 1}(1 - z_1)(1/z_1!)(\frac{1}{2})^{z_1+1}\mathrm{e}^{-1}\alpha_3^{-1} \geqslant -\frac{1}{8}\mathrm{e}^{-1}\alpha_3^{-1}$,
- $\mathrm{con}_4 = -\sum_{z_1 \geqslant 0}(1 + z_1)(1/z_1! \cdot 2!)(\frac{1}{2})^{z_1+2}\mathrm{e}^{-1}\alpha_3^{-1} \geqslant -\frac{1}{3}\mathrm{e}^{-1}\alpha_3^{-1}$,
- $\mathrm{con}_5 = -\frac{3}{48}\mathrm{e}^{-1}\alpha_3^{-1}$,
- $\mathrm{con}_6 \geqslant 0$,

as contributions for the subcase $s = 3$. Obviously, $E(D_S^*(x))$ is bounded below by a positive constant also in this last subcase.

Altogether, $D_S^*(x)$ has all desired properties and, in particular, $E(D_S^*(x)) \geqslant d^*$ for some positive constant $d^*$ if $n$ is large enough.

Let $T^*(x)$ be the random variable describing the first point of time where the value of the bit string produced by successful steps of the $(1 + 1)$ EA on the linear function $f$ starting at $x$ is larger than $\mathrm{val}(x)$. We claim that $E(T^*(x)) \leqslant c^* := 1/d^*$. We prove this claim by considering the changes of val. We investigate the stochastic process related to the (1+1) EA on $f$ where the initial string is $X_0 = x$. Let $X_0, X_1, X_2, \ldots$, be the sequence of actual strings created by successful steps. Then we look for the first point of time $t$ where $Y_t := \mathrm{val}(X_t) - \mathrm{val}(X_0)$ takes a positive value. The sequence $Y_0, Y_1, Y_2, \ldots$ is a stochastic process on $\mathbb{Z}$. The distribution of $Y_{t+1} - Y_t$ for some given value of $X_t = x'$ (and, therefore, also $Y_t$) is described by the random variable $D(x')$ which equals $D_S(x')$ if $S$ is the set of positions flipping from 0 to 1. We have shown that we obtain an upper bound on $T^*(x)$ if we replace $D(x')$ with $D^*(x')$ (and $D_S(x')$ with $D_S^*(x')$). Since $D^*(x) \leqslant 1$, the first positive value reached by this new process is the point 1. We cannot apply Wald's identity, since $D^*(x')$ depends on $x'$. Therefore, we need a slight generalization of Wald's identity. We prove the claim by induction on $\mathrm{val}(x)$. The base of induction, $\mathrm{val}(x) = 0$, considers the all-zero string and is obvious, since each successful step leads to a positive value of val. Let the claim be true for all $x'$ where $\mathrm{val}(x') < k$ and let $x$ be one of the strings where $\mathrm{val}(x) = k$ and where among these strings $E(T^*(x))$ is maximal. If $D^*(x) = 1$, we are done after one step. If $D^*(x) = d \leqslant 0$, we need by the induction hypothesis on average at most $-dc^*$ steps until we reach 0 again. Moreover, we need at most an expected number of $E(T^*(x))$ steps to reach 1.

Hence,

$$\begin{aligned} E(T^*(x)) &\leqslant 1 + \text{Prob}(D^*(x) \neq 1) \cdot E(T^*(x)) - \sum_{d \leqslant 0} \text{Prob}(D^*(x) = d) dc^* \\ &= 1 + \text{Prob}(D^*(x) \neq 1) \cdot E(T^*(x)) - \sum_{d \leqslant 1} \text{Prob}(D^*(x) = d) dc^* \\ &\quad + c^* \,\text{Prob}(D^*(x) = 1) \\ &= 1 + E(T^*(x)) - E(T^*(x)) \,\text{Prob}(D^*(x) = 1) - E(D^*(x)) c^* \\ &\quad + c^* \,\text{Prob}(D^*(x) = 1). \end{aligned}$$

This implies

$$\begin{aligned} E(T^*(x)) \,\text{Prob}(D^*(x) = 1) &\leqslant 1 - E(D^*(x)) c^* + c^* \,\text{Prob}(D^*(x) = 1) \\ &\leqslant c^* \,\text{Prob}(D^*(x) = 1). \end{aligned}$$

This last inequality follows, since $E(D^*(x)) \geqslant d^* = 1/c^*$. Since $\text{Prob}(D^*(x) = 1) > 0$, we obtain the desired result $E(T^*(x)) \leqslant c^*$. This proves the theorem. □

We see that the $(1 + 1)$ EA optimizes linear functions quite efficiently. One may ask whether the performance depends on the mutation probability $p_m$, whether the performance can be improved substantially by using other values than $1/n$ for $p_m$.

The "correct" mutation probability has already been subject to some research, see, e.g., [2]. The choice of $p_m = 1/n$ is the most often recommended one, but the reasoning is basically based on experimental experience. We prove here that mutation probabilities of $c/n$ for positive constants $c$ are optimal for linear functions and that much smaller or larger probabilities increase the expected running time.

For much smaller mutation probabilities, we expect that the waiting time until all bits have tried to flip becomes too long. For much larger mutation probabilities, in each step on average a lot of bits try to flip, so we expect the probabilities of successful steps may become too small. We justify this reasoning by two formal statements.

**Theorem 13.** *The* $(1+1)$ *EA with mutation probability* $p_m = (\alpha(n)n)^{-1}$, *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 linear function with positive weights.*

**Proof.** With probability at least $1/2$ we have at least $n/2$ zeros in the initial bit string. The following analysis works under this condition. As in the proof of Lemma 10 let $T$ be the random variable describing the first point of time where each of the $n/2$ zero bits has tried to flip. If $\text{Prob}(T \geqslant t) \geqslant c$ for some constant $c$ and all $t \leqslant \alpha(n)n \ln n - \ln n$, then the theorem follows. Let $t = (n\alpha(n) - 1) \ln n$. Then $(1 - 1/n\alpha(n))^t \geqslant \mathrm{e}^{-\ln n} = 1/n$

and

$$\text{Prob}(T \geqslant t+1) = 1 - \left(1 - \left(1 - \frac{1}{n\alpha(n)}\right)^t\right)^{n/2} \geqslant 1 - \left(1 - \frac{1}{n}\right)^{n/2}$$

$$\geqslant 1 - \mathrm{e}^{-1/2}. \qquad \square$$

**Theorem 14.** *The* $(1+1)$ *EA with mutation probability* $p_m = \alpha(n)/n$, *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* ONEMAX.

**Proof.** The probability that the initial string has less then $n/3$ ones is exponentially small. Otherwise, we always have at least $n/3$ ones. None of these ones flips during the last step creating the optimal all-one string. Hence, the expected running time in this case is at least the reciprocal value of the probability that $n/3$ specified bits do not flip. Hence, we get the lower bound

$$(1 - \mathrm{o}(1))\left(1 - \frac{\alpha(n)}{n}\right)^{-n/3} \geqslant \mathrm{e}^{c\alpha(n)}$$

for some constant $c > 0$. Since $\alpha(n)n\ln n \leqslant \mathrm{e}^{3\ln n}$, we get the claimed bound if $\alpha(n) \geqslant 3c^{-1}\ln n$.

If $\alpha(n) = \mathrm{O}(\log n)$, we only investigate the time interval $I$ starting with the first point of time where the current string $x^*$ has at least $n - n^{1/2}$ ones. Let $A$ be the event that $x^*$ has at most $n - n^{1/2}/4$ ones. We prove the bound by showing that $\text{Prob}(A) = 1 - \mathrm{o}(1)$ and that the expected running time given $A$ is bounded by $\Omega(\alpha(n)n\ln n)$.

First, we prove that $\text{Prob}(A) = 1 - \mathrm{o}(1)$. The probability that the initial string has more than $n - n^{1/2}$ ones is exponentially small. Since $p_m = \mathrm{O}((\log n)/n)$, the probability that less than a fraction of $3/4$ of the former zeros flips is $1 - \mathrm{o}(1)$.

Now we estimate the expected running time given $A$. The probability of at least five flipping zeros is bounded above by

$$\binom{n^{1/2}}{5}\left(\frac{\alpha(n)}{n}\right)^5 = \mathrm{O}((\log n)^5 n^{-5/2}).$$

If such a step happens within $\alpha(n)n\ln n$ steps, we estimate the running time by 0. Otherwise, i.e., with probability $1 - \mathrm{o}(1)$, the probability of increasing the number of ones during a step with a current string with $N$ zeros is

$$\sum_{0 \leqslant j < i \leqslant 4} \binom{N}{i} p_m^i \binom{n-N}{j} p_m^j (1 - p_m)^{n-i-j}$$

$$\leqslant \sum_{0 \leqslant j < i \leqslant 4} N^i n^j \left(\frac{\alpha(n)}{n}\right)^{i+j} \mathrm{e}^{-\Omega(\alpha(n))} = Nn^{-1}\alpha(n)^7 \mathrm{e}^{-\Omega(\alpha(n))}.$$

Hence, the expected waiting time to increase the number of ones is

$$(1/N)n\alpha(n)^{-7}e^{\Omega(\alpha(n))} = (1/N)\Omega(\alpha(n)\cdot n).$$

Since the number of ones is increased by our assumptions in each step at most by an additive term of 4, the expected running time of the $(1+1)$ EA is at least

$$\left(\frac{1}{4}+\frac{1}{8}+\frac{1}{12}+\cdots+\frac{1}{r}\right)\Omega(\alpha(n)n),$$

where $r$ is the largest multiple of 4 smaller than $n^{1/2}/4$ and, therefore, the lower bound is $\Omega(\alpha(n)n\ln n)$. □

## 4. Unimodal functions

Linear functions are our first example of a class of functions where the $(1+1)$ EA is expected to find the global optimum quite efficiently. Of course, we would like to identify more and larger classes of functions where some polynomial upper bound on the expected running time can be given. Since we know from Section 2 that already functions with degree 2 can be most difficult for the $(1+1)$ EA, we need some other criterion to recognize a fitness function as easy.

In this section we consider unimodal functions. Since we are considering fitness functions with Boolean inputs, it is not totally obvious how "unimodal" should be defined. In fact, there are different definitions in the literature which yield quite different classes of functions. Here, we use that definition which appears to be the most natural one.

**Definition 15.** Let $f:\{0,1\}^n\to\mathbb{R}$ be a fitness function. We call $x\in\{0,1\}^n$ a *local maximum*, iff

$$\forall y\in\{0,1\}^n : H(x,y)=1 \Rightarrow f(y)\leqslant f(x).$$

The function $f$ is *unimodal* iff $f$ has exactly one local maximum.

Obviously, all linear functions with all nonzero weights are unimodal. Moreover, the most striking similarity between unimodal and linear functions is that for all $x\in\{0,1\}^n$ that are not the global optimum there is always another point with Hamming distance 1 with greater fitness. So there is always a mutation of exactly one bit that improves the function value. This leads Mühlenbein [15] to the remark that all unimodal functions can be optimized by the $(1+1)$ EA in expected $O(n\log n)$ steps without clearly defining his notion of unimodality. We take unimodal to be defined as above and disprove this claim in this section.

### 4.1. A quadratic lower bound for a unimodal fitness function

Already Rudolph [17] doubts Mühlenbein's claim and presents a fitness function and believes that the $(1+1)$ EA has expected running time $\Theta(n^2)$ on that function. He proves an upper bound of $O(n^2)$ steps and presents experiments that confirm the lower bound, but does not give a formal proof. We use his example and present the lower bound.

**Definition 16.** The function LeadingOnes: $\{0,1\}^n \to \mathbb{R}$ is defined by

$$\text{LeadingOnes}(x_1,\ldots,x_n) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

The function value of LeadingOnes$(x)$ equals the number of leading ones in $x$. Since the function value can always be increased by appending a single one to the leading ones, LeadingOnes is obviously unimodal. The $(1+1)$ EA accepts a mutation iff the number of leading ones is not decreased by this mutation.

**Theorem 17.** *The expected running time for the* $(1+1)$ *EA on the function* LeadingOnes *is* $\Theta(n^2)$. *Moreover, there are constants* $c_1, c_2 > 0$ *such that the probability that the running time of the* $(1+1)$ *EA on* LeadingOnes *is outside the interval* $[c_1 n^2, c_2 n^2]$ *is exponentially small.*

**Proof.** We repeat the easy proof of the upper bound. If $x$ is not optimal, the number of leading ones is increased iff the first zero in $x$ flips and no bit among the leading ones flips. The probability of such a success is bounded below by $1/n(1-1/n)^{n-1} \geqslant e^{-1}/n$. It is obvious that we have reached the optimum after at most $n$ successes. The expected time for $n$ successes is bounded above by $en^2$. By Chernoff's bounds, the probability of not having reached the optimum after $2en^2$ steps is bounded above by $\exp(-n/4)$.

For the lower bound we investigate the stochastic process behind the $(1+1)$ EA. Let $i$ be the number of leading ones of the current string $x$. If $i=n$, we are done. If $i<n$, $x_{i+1}=0$. We claim that $(x_{i+2},\ldots,x_n)$ is a random string uniformly distributed over $\{0,1\}^{n-i-1}$. This obviously holds for the initial string. We assume that it holds for the current string $x$. The new string $x'$ is accepted iff the first $i$ bits have not flipped. It is obvious that the random mutation of a random string leads to a random string, i.e., $(x'_{i+2},\ldots,x'_n)$ also is random. If the parameter $i$ changes, it increases and the suffix after the first 0 is still random.

Let $i$ be the number of leading ones of the current string $x$. The probability that we increase the number of leading ones during the next step is bounded above by $1/n$, since it is necessary that the bit at position $i+1$ flips. Such steps are called *essential*. We have $k$ "free-riders", if exactly the $k$ leading bits of the string $(x'_{i+2},\ldots,x'_n)$ are ones. Since this is a random string the expected number of free-riders is less than 1.

We prove that with high probability $n^2/6$ steps are not sufficient to reach the optimum for LeadingOnes. We consider two types of "failures":
– there are at least $n/3$ essential steps,
– there are more than $2n/3$ free-riders during $n/3$ occasions.
If we have less than $n/3$ essential steps, we have at most $n/3$ occasions where we may have free-riders. If the number of free-riders is bounded by $2n/3$ and the number of essential steps is less than $n/3$, the optimization process is not finished. We use Chernoff's bounds to estimate the failure probability. The first failure probability is bounded above by $(e/4)^{n/6}$ and is exponentially small. For the second failure, we consider a random 0–1-string of arbitrary length with the following interpretation. The number of ones between the $(i-1)$th and the $i$th zero, where $i \geqslant 1$, is the number of free-riders during the $i$th occasion. We have seen above that the random number of leading ones of an initial string describes the random number of free-riders (besides the fact that our string has bounded length $n$). If and only if the random string has less than $n/3$ zeros among the first $n$ positions, the number of free-riders during $n/3$ occasions is larger than $2n/3$. The probability of this event is bounded above by $\exp(-n/36)$ and is exponentially small. Hence, the probability that the number of steps is at least $n^2/6$ is exponentially close to 1 and this implies the lower bound on the expected time. □

### 4.2. An exponential lower bound for a unimodal fitness function

It is obvious that the $(1+1)$ EA reaches the optimum of a unimodal function with $k$ different function values after an expected number of $O(kn)$ steps [17]. For unimodal functions there is always at least one possible mutation that increases the function value and requires only the mutation of exactly one bit. Since such mutations have probability $(1/n)(1-1/n)^{n-1}$, the expected time until such a mutation occurs is bounded above by $en$. At most $k$ such mutations are sufficient, so the bound $O(kn)$ follows. To enforce large expected running times on unimodal functions one needs a fitness function where only a large number of 1 bit mutations is sufficient to reach the optimum by such "small steps". The idea is to construct functions where only a "path" to the optimum exists: a path is a sequence of bit strings that are all reachable via mutations of exactly one bit such that this mutation is the only mutation of exactly one bit that is accepted. If this path is exponentially long, one may hope that the $(1+1)$ EA needs exponentially many steps to find the optimum in the expected case.

Such long paths were introduced by Horn et al. [9]. They performed several experiments and compared the $(1+1)$ EA and some other hillclimbers with a Genetic Algorithm. Though they were convinced to observe exponential running times of the $(1+1)$ EA, Rudolph [18] proved that the expected running time is only $O(n^3)$. The problem is that already a mutation of two bits simultaneously enables the $(1+1)$ EA to take a shortcut and this reduces the number of required steps dramatically. To overcome this problem Rudolph [17] formally defines a variant of long paths (already informally described by Horn et al. [9], such that no mutation of at most $k$ bits is sufficient to take a short cut. These paths are called long $k$-paths. The parameter $k$ can

be chosen, though with increasing $k$ the length of the path decreases. We start with defining long $k$-paths and a few simple statements about them that are taken from [17].

**Definition 18.** Let $n \geqslant 1$ hold. For all $k>1$ where $(n-1)/k \in \mathbb{N}$ the long $k$-path of dimension $n$ is a sequence of bit strings from $\{0,1\}^n$. The long $k$-path of dimension 1 is defined as $P_1^k := (0,1)$. The long $k$-path of dimension $n$ is defined using the long $k$-path of dimension $n-k$ as basis as follows. Let the long $k$-path of dimension $n-k$ be given by $P_{n-k}^k = (v_1,\ldots,v_l)$. Then we define the sequences of bit strings $S_0$, $B_n$, and $S_1$ from $\{0,1\}^n$, where $S_0 := (0^k v_1, 0^k v_2, \ldots, 0^k v_l)$, $S_1 := (1^k v_l, 1^k v_{l-1}, \ldots, 1^k v_1)$, and $B_n := (0^{k-1}1v_l, 0^{k-2}11v_l, \ldots, 01^{k-1}v_l)$. The points in $B_n$ build a bridge between the points in $S_0$ and $S_1$, that differ in the $k$ leading bits. Therefore, the points in $B_n$ are called *bridge points*. The resulting long $k$-path $P_n^k$ is constructed by appending $S_0, B_n$, and $S_1$, so $P_n^k$ is a sequence of $|P_n^k| = |S_0| + |B_n| + |S_1|$ points. We call $|P_n^k|$ the length of $P_n^k$. The $i$th point on the path $P_n^k$ is denoted as $p_i$; $p_{i+j}$ is called the $j$th successor of $p_i$.

The recursive definition of long $k$-paths allows us to determine the length of the paths easily. We remark that it is for our purpose more convenient to define the length of a path by the number of nodes on the path and not (as usual) by the number of edges.

**Lemma 19.** *The long $k$-path of dimension $n$ has length $|P_n^k| = (k+1)2^{(n-1)/k} - k + 1$. All points of the path are different.*

**Proof.** For $n=1$ the length is $(k+1)2^0 - k + 1 = 2$. Let the statement hold for values smaller than $n$. By definition of long $k$-paths we have $|P_n^k| = 2|P_{n-k}^k| + k - 1 = 2(k+1)2^{(n-k-1)/k} - 2k + 2 + k - 1 = (k+1)2^{(n-1)/k} - k + 1$. By definition all points on the path are different. □

The most important property of long $k$-paths is the simple rule that holds for the Hamming distances between each point and its successors.

**Lemma 20.** *Let $n$ and $k$ be given such that the long $k$-path $P_n^k$ is well defined. For all $i$ with $0<i<k$ the following holds. If $x \in P_n^k$ has at least $i$ different successors on the path, the $i$th successor of $x$ has Hamming distance $i$ of $x$ and all other points on the path that are successors of $x$ have Hamming distances different from $i$.*

**Proof.** The statement is obviously true for $n=1$ and all values of $k$. Assume that it holds for $P_{n-k}^k$. We know that $P_n^k$ is constructed by appending $S_0, B_n$, and $S_1$, with $|P_n^k| = (k+1)2^{(n-1)/k} - k + 1$, $|S_0| = |S_1| = (k+1)2^{(n-k-1)/k} - k + 1$, and $|B_n| = k - 1$. Let $x$ be the $p$th point of $P_n^k$. We distinguish several cases according to the value of $p$.

If $p < |S_0|$ and $p + i \leqslant |S_0|$, the Hamming distance between $x$ and its $i$th successor on the path equals $i$ by the induction hypothesis. The last point belonging to $S_0$ has by induction hypothesis a Hamming distance of at least $i$ from $x$. Hence, the Hamming

distance between $x$ and points on the bridge is larger than $i$ and all points from $S_1$ have a Hamming distance of at least $k>i$ from the points of $S_0$.

If $p \leqslant |S_0|$ and $|S_0| < p+i \leqslant |S_0|+|B_n|$, the Hamming distance from $x$ to the last point in $S_0$ is $|S_0|-p$ which is at least 0 and less than $k$ by the assumption $0<i<k$. By definition of $B_n$, the $j$th point in $B_n$ differs from the last point of $S_0$ in $j$ bits, so there is exactly one point in $B_n$ with Hamming distance $i$. All points in $S_1$ have greater Hamming distance, since the first $k$ bits of points in $S_0$ and $S_1$ are all different.

If $|S_0| < p < |S_0|+|B_n|$ and $p+i \leqslant |S_0|+|B_n|$, the statement is obviously true. All points in $B_n$ just differ on the first $k$ bits, the Hamming distance of any point to its $j$th successor is $j$ so $x$ has exactly one successor in $B_n$ with Hamming distance $i$. All points in $S_1$ have greater Hamming distance.

If $|S_0| < p \leqslant |S_0|+|B_n|$ and $|S_0|+|B_n| < p+i$ hold, let $j = |S_0|+|B_n|-p+1$. The $(p+j)$th point $x'$ on the path has a Hamming distance of $j$ from $x$ and a Hamming distance of $i-j$ from the $i$th successor of $x$ on the path. The result follows, since the Hamming distance of each point $x''$ from $S_1$ to $x$ is by an additive term $j$ larger than the Hamming distance of $x''$ to $x'$. This implies the statement by the induction hypothesis.

Finally, if $|S_0|+|B_n| < p$, the statement holds by assumption, since $S_1$ has the same structure as $P^k_{n-k}$. □

We are interested in unimodal fitness functions, so for a fitness function $f: \{0,1\}^n \to \mathbb{R}$ exactly $2^n$ function values have to be defined. Since the long $k$-path of dimension $n$ consists of only $(k+1)2^{(n-1)/k}-k+1$ points, we have to embed this path in a unimodal function. The following definition differs from the one given by Rudolph [17] in the way bit strings not belonging to the long $k$-path are treated. This little modification simplifies the lower bound proof while it does not matter for the upper bounds that were already given by Rudolph.

**Definition 21.** Let $n \geqslant 1$ hold, let $k>1$ be given such that $k$ satisfies $(n-1)/k \in \mathbb{N}$. The long $k$-path function of dimension $n$ is called $\textsc{Path}_k$: $\{0,1\}^n \to \mathbb{N}$ and is defined by

$$\textsc{Path}_k(x) = \begin{cases} n^2 + l & \text{if } x \text{ is the } l\text{th point of } P^k_n, \\ n^2 - n\sum_{i=1}^{k} x_i - \sum_{i=k+1}^{n} x_i & \text{if } x \notin P^k_n. \end{cases}$$

We have already mentioned that a fitness function $f$ is unimodal iff for all points $x \in \{0,1\}^n$ either $x$ is the global maximum or there exists $y \in \{0,1\}^n$ with $H(x,y)=1$, such that $f(x)<f(y)$ holds. For $\textsc{Path}_k$ this is obviously the case. For all points on the path except the last one, which is the global optimum, this holds, since there is always one successor on the path with Hamming distance exactly 1 according to Lemma 20. For points not on the path decreasing the number of ones by 1 always yields a bit string with increased function value. By Definition 18 it is obvious that the all-zero bit string is the first point on the path.

Rudolph [17] establishes two upper bounds on the expected running time that both yield exponential values for $k=\sqrt{n-1}$, so he speculates that the expected running time may in fact be exponential for this choice of $k$. We prove this here, and thereby answer the open question, whether unimodal functions exist, on which the $(1+1)$ EA has exponential expected running time.

**Lemma 22** (Rudolph[17]). *The expected running time of the* $(1+1)$ *EA on* $\textsc{Path}_k$ *is bounded above by* $O(n^{k+1}/k)$ *and* $O(n|P_n^k|)$.

**Proof.** During a run of the $(1+1)$ EA the function values of the current string never decrease. We divide a run of the algorithm into two different phases. In the first phase after random initialization the long $k$-path is reached. Note, that once the path is reached it cannot be left again. In the second phase the global optimum of $P_n^k$ is found. We denote the number of steps in the first phase by $T_1$ and in the second phase by $T_2$. We prove an upper bound on $E(T_1)$ and two upper bounds on $E(T_2)$.

The first phase starts with some point that does not belong to the path. Otherwise $T_1=0$. The first phase ends when a point on the path is reached for the first time. The function $\textsc{Path}_k$ restricted to the points not on the path is linear with weights $w_1=\cdots=w_k=-n$, $w_{k+1}=\cdots=w_n=-1$, and the constant additive term $\tau=n^2$ (see Eq. (1)). Moreover, all points on the path have larger fitness. Hence, the expected time to reach a point on the path is not larger than the expected time to optimize the described linear function. We conclude from Theorem 12 that $E(T_1)=O(n\log n)$ holds. Since $k>1$, $n\log n=O(n^{k+1}/k)$. If $k=n-1$, we have $|P_n^k|=2n-(n-1)+1=\Omega(n)$. Otherwise, we have $k\leqslant(n-1)/2$. The long $k$-path of dimension $n$ $P_n^k$ is constructed recursively based on $P_{n-k}^k$. In each construction step a "bridge" of length $k-1$ is inserted. There are $((n-1)/k)-1$ recursive construction steps. Thereby, we have $|P_n^k|\geqslant((n-k-1)/k)(k-1)=\Omega(n)$ in this case. Hence, the length of $P_n^k$ is always $\Omega(n)$ and $n\log n=O(n|P_n^k|)$.

In order to estimate $E(T_2)$ we use the following approach. The path $P_n^k=(a_1,a_2,\ldots,a_l)$ where $l=|P_n^k|$ is divided into $r$ consecutive subpaths $U_1,\ldots,U_r$ where $U_r=\{a_l\}$ only consists of the global optimum. Let $p^*$ be a lower bound on the probability of reaching a point in $U_{i+1}\cup\cdots\cup U_r$ from a point in $U_i$ for all $i$. This implies the upper bound $(r-1)/p^*$ on $E(T_2)$, since from $U_i$ only steps to $U_i\cup\cdots\cup U_r$ are accepted.

For the first upper bound set $r=l$ and $U_i=\{a_i\}$. Since $a_{i+1}$ has Hamming distance 1 from $a_i$, $p^*\geqslant(1/n)(1-1/n)^{n-1}\geqslant 1/(en)$ in this case and $E(T_2)=O(n|P_n^k|)$.

For the second upper bound set $r=(n-1)/k+2$. The path $P_n^k$ is constructed from two "copies" of $P_{n-k}^k$ denoted by $S_0$ and $S_1$ and a bridge between them. Let $U_1$ contain $S_0$ and the bridge. Then we follow this approach to construct $U_1,\ldots,U_r$ for $S_1$. At the end we obtain a path of length 2 and $U_{r-1}$ and $U_r$ contain the points of this path. The crucial observation is that for each point $a$ in $S_0$ or the bridge there is a point $b$ in $S_1$ such that the Hamming distance between $a$ and $b$ is at most $k$. Hence, $p^*\geqslant(1/n)^k(1-1/n)^{n-k}\geqslant 1/(en^k)$ in this case and $E(T_2)=O(n^{k+1}/k)$. □

**Theorem 23.** *The expected running time of the* (1+1) *EA on* $\textsc{Path}_{\sqrt{n-1}}$ *is* $\Theta(n^{3/2}2^{\sqrt{n}})$.

**Proof.** For $k=\sqrt{n-1}$ we have $|P_n^k|=(k+1)2^k-k+1$ according to Lemma 19, so the upper bound follows directly from Lemma 22.

For the lower bound we reconsider the first phase until a point $x^*$ on $P_n^k=(a_1,\ldots,a_l)$ is reached and the second phase until the optimum is reached. For the first phase we use the trivial lower bound 0 for the running time. We claim that the probability that $x^*$ belongs to the first part $S_0$ of $P_n^k$ or to the bridge $B_n$ is at least $1/4$ (this bound can be improved to $1/2$). For this reason we denote the points of $S_0$ by $0^kv_1,\ldots,0^kv_l$. By definition, $S_1$ contains the points $1^kv_l,\ldots,1^kv_1$. The probability that the initial bit string $x$ contains at least $k/2$ zeros among the first $k$ bits is at least $1/2$. In the following we assume that this event has occurred. As long as no point of $P_n^k$ is reached, the definition of $\textsc{Path}_k$ implies that the number of zeros at the first $k/2$ positions does not decrease. Hence, the probability of creating $0^kv_i$ by mutation is always at least as large as the probability of creating $1^kv_{l-i+1}$. It remains to prove the lower bound $\Omega(n^{3/2}2^{\sqrt{n}})$ on the expected running time under the assumption that we start from a point $a_i$ belonging to $S_0$ or $B_n$.

Let $a_j$ be an arbitrary point from $P_n^k$. Only the points $a_j,\ldots,a_l$ have at least the same fitness as $a_j$. Hence, after one step we have reached some point $a_m$ where $m\geqslant j$. The progress caused by this step is measured by $m-j$ and we claim (for large $n$) that the expected progress can be estimated by $2/n$ (independently of $j$). Here we use the facts contained in Lemma 20. With probability $(1/n)^r(1-1/n)^{n-r}\leqslant(1/n)^r$ the mutation of $a_j$ results in $a_{j+r}$ if $j+r\leqslant l$ and $1\leqslant r\leqslant k-1$. In order to reach one of the points $a_{j+k},\ldots,a_l$ at least $k$ bits have to flip. The probability for this event is bounded above by $\binom{n}{k}(1/n)^k\leqslant 1/k!$. In this case the progress is bounded by the trivial bound $|P_n^k|$. In all other cases the progress is zero. Hence, the expected progress can be estimated by

$$\sum_{r=1}^{k-1}\frac{r}{n^r}+\frac{1}{k!}|P_n^k|.$$

Using the formulas for geometric series we obtain

$$\sum_{r=1}^{k-1}\frac{r}{n^r}<\sum_{r=1}^{\infty}\sum_{j=r}^{\infty}\frac{1}{n^j}=\frac{n}{(n-1)^2}=\frac{1}{n-1}+\frac{1}{(n-1)^2}.$$

Using Lemma 19 and the fact $k=\sqrt{n-1}$ we obtain that

$$|P_n^k|\leqslant(k+1)2^{(n-1)/k}=2^{O(\sqrt{n})}$$

and, by Stirling's formula,

$$k!=2^{\Omega(\sqrt{n}\log n)}.$$

Hence, the expected progress can be estimated by

$$\frac{1}{n-1}+\frac{1}{(n-1)^2}+2^{-\Omega(\sqrt{n}\log n)}$$

which is at most $2/n$ for large $n$.

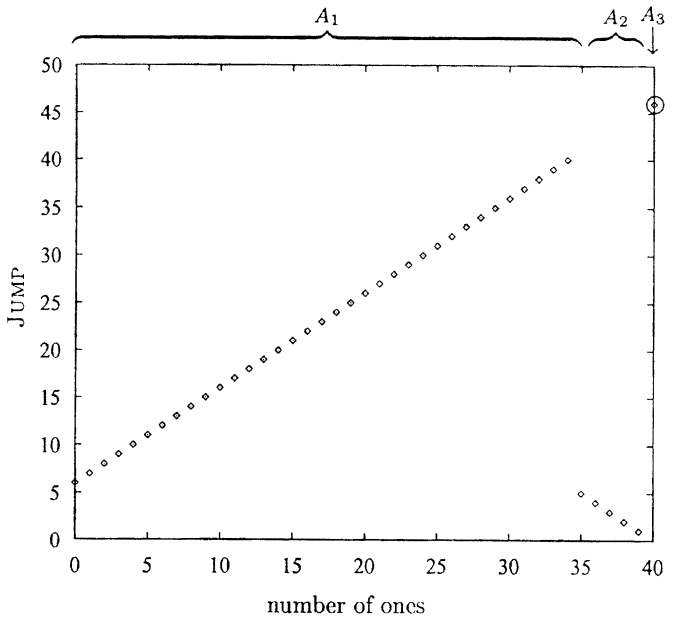


Fig. 2. The function $\text{JUMP}_{m,n}$ for $n=40$ and $m=6$.

Under the condition that $x^*$ does not belong to $S_1$, the necessary progress is at least $|S_1| = |P^k_{n-k}| \geqslant k2^{(n-k-1)/k} \geqslant c\sqrt{n}2^{\sqrt{n}}$ for some constant $c>0$. The expected progress within $(1/4)cn^{3/2}2^{\sqrt{n}}$ steps is bounded above by $(1/2)c\sqrt{n}2^{\sqrt{n}}$ and, by Markoff's inequality, the probability that the global optimum is reached within $(1/4)cn^{3/2}2^{\sqrt{n}}$ steps is bounded above by $1/2$. Altogether, we have proved the proposed lower bound. □

## 5. Enforcing expected running times

We have a class of fitness functions which the $(1+1)$ EA optimizes efficiently, and we know several examples where the expected running time of the $(1+1)$ EA is exponential. In this section we define $n$ different fitness functions $\text{JUMP}_{1,n},\ldots,\text{JUMP}_{n,n}$, where the expected running time is $\Theta(n^m + n\log n)$ for $\text{JUMP}_{m,n}$. So we can enforce a large variety of different expected running times. This result can be interpreted as a hierarchy result for the performance of the $(1+1)$ EA. Each additional factor $n$ for a permitted upper bound on the expected time enlarges the class of functions which can be optimized. The construction of the functions $\text{JUMP}_{m,n}$ is done in a way that shows the understanding of the way Algorithm 1 works. The main idea is to use a construction similar to the TRAP-function, but to adjust the distance between the local and the global maximum, so that the expected running time, that is dominated by the time for the jump over that distance, can be controlled. The function $\text{JUMP}_{m,n}$, which is given in the following definition, is visualized in Fig. 2 for $n=40$ and $m=6$. With

growing $m$ the size of the gap widens, for $m = n$ the function $\textsc{Jump}_{n,n}$ equals $\textsc{Trap}$. For $m = 1$ we get the linear function $\textsc{OneMax}$.

**Definition 24.** Given $n$ with $n > 1$ and $m \in \{1, 2, \ldots, n\}$, let the function $\textsc{Jump}_{m,n}$: $\{0,1\}^n \to \mathbb{R}$ be defined by

$$JUMP_{m,n}(x) := \begin{cases} m + \sum_{i=1}^{n} x_i & \text{if } \sum_{i=1}^{n} x_i \leqslant n - m \text{ or } \sum_{i=1}^{n} x_i = n, \\ n - \sum_{i=1}^{n} x_i & \text{otherwise.} \end{cases}$$

**Theorem 25.** *The expected running time of the* $(1+1)$ *EA on* $\textsc{Jump}_{m,n}$ *is* $\Theta(n^m + n \log n)$ *for* $m \in \{1, 2, \ldots, n\}$.

**Proof.** In this proof we omit the subscript $n$ in $\textsc{Jump}_{m,n}$. As we mentioned above, for $m = 1$ the function $\textsc{Jump}_1$ essentially equals $\textsc{OneMax}$ (in fact it gives $\textsc{OneMax}+1$), so the expected running time $\Theta(n \log n)$ follows from the results of Section 3.

Now we assume that $m > 1$ holds. We partition the search space into three disjoint sets $A_1$, $A_2$, and $A_3$ with

$$A_1 := \left\{ x \in \{0,1\}^n \mid n - m < \sum_{i=1}^{n} x_i < n \right\},$$

$$A_2 := \left\{ x \in \{0,1\}^n \mid \sum_{i=1}^{n} x_i \leqslant n - m \right\}$$

$$A_3 := \{x_{\text{one}}\}.$$

The definition of the sets is done in a way that ensures that bit strings in $A_i$ have greater function values than bit strings in $A_j$, iff $i > j$ holds. So after reaching some bit string in $A_j$ the $(1+1)$ EA can only reach bit strings in $A_i$ with $i \geqslant j$. Since $\textsc{Jump}_m$ is a symmetric function, we can ignore steps where the number of ones in the bit string remains unchanged.

We begin with an upper bound on the expected running time. The expected number of steps until the $(1+1)$ EA reaches either a bit string with exactly $m$ zeros or alternatively the global optimum is $O(n \log n)$. If the current bit string belongs to $A_1$, steps towards bit strings with more zeros are accepted. The situation is similar to the linear function $f(x) = -\sum_{i=1}^{n} x_i$, except for steps to the global optimum, which are, of course, accepted. We conclude that with high probability after $O(n \log n)$ steps either the global maximum or a bit string from $A_2$ is reached. So we now assume that the current bit string belongs to $A_2$. The situation is similar to $\textsc{OneMax}$, as long as the number of zeros in the current bit string is at least $m$. We conclude that again after $O(n \log n)$ steps either the global maximum or a bit string with exactly $m$ zeros is reached. Once the $(1+1)$ EA reaches a bit string with exactly $m$ zeros, only mutations to other bit strings with exactly $m$ zeros and mutations to the global maximum are accepted. The probability for reaching the global maximum by an $m$-bit mutation

equals $n^{-m}(1-1/n)^{n-m} \geqslant e^{-1}n^{-m}$. We conclude that we have an upper bound of $O(n^m + n\log n)$ on the expected running time.

Now we derive the lower bound. Let $T$ denote the number of steps until the optimum is reached. Let $q$ be the probability that some bit string in $A_2$ is reached, then we claim that $E(T) \geqslant qn^m$.

If we have reached $x \in A_2$, we never accept a string from $A_1$. Hence, the global optimum has to be reached directly by a mutation of some $y \in A_2$. The probability to obtain the global optimum from $y \in A_2$ is bounded above by $n^{-m}(1-1/n)^{n-m} \leqslant n^{-m}$. Hence, with probability $q$ the expected running time is at least $n^m$.

The probability that the initial bit string contains at least $n/2$ zeros is at least $1/2$. It remains to prove that the probability of reaching some bit string in $A_2$ if we start with a bit string with at least $n/2$ zeros is bounded below by a positive constant $\varepsilon$. If $m \leqslant n/2$, the initial bit string belongs to $A_2$. Hence, we can assume that $m > n/2$. We consider the first $n^2$ steps. The number of ones only gets larger than $n/2$ if we reach the global optimum. The probability for such a jump is bounded above by $n^{-n/2}$ and, therefore, the probability of reaching the global optimum within $n^2$ steps is exponentially small. By Markoff's inequality the probability of reaching a string of $A_2$ within $n^2$ steps is at least $1-O((n\log n)/n^2) = 1-O((\log n)/n)$. This proves the lower bound. □

## 6. A variant of the (1 + 1) EA

When we take a closer look at the $(1+1)$ EA, i.e. Algorithm 1, we see that the old bit string $x$ is replaced by a new bit string $x'$, even if $f(x) = f(x')$. This strategy of *accepting equal-valued bit strings* is often used in many Evolutionary Algorithms, as it is assumed to help the algorithm to escape from plateaus, i.e. sets of neighboring bit strings with equal fitness value: if the actual bit string is "surrounded" by bit strings of the same fitness value and Algorithm 1 would only accept bit strings with higher fitness, it would need a long time to make an improvement. By accepting bit strings with the same fitness, the $(1+1)$ EA can make random steps on this plateau, which can bring it nearer to bit strings with higher fitness, therefore making it more likely to escape from this plateau.

Now we will show that this common-sense argumentation for the strategy of accepting equal-valued bit strings can be rigorously proven for the PEAK-function to lower the growth of the expected running time of the $(1+1)$ EA substantially. The PEAK-function is defined by

$$\text{PEAK}(x_1,\ldots,x_n) := \prod_{i=1}^{n} x_i.$$

The PEAK-function should be well suited for our purpose, as it has only one peak, while all other bit strings form one big plateau, therefore giving no "hints" about the peak.

**Theorem 26.** *The* $(1+1)$ *EA has an expected running time bounded above by* $2^n(1+o(1))$ *for* PEAK. *If the* $(1+1)$ *EA only accepts bit strings with higher fitness, the expected running time for* PEAK *is bounded below by* $e^{n\ln(n/2)}$.

**Proof.** The upper bound for the original $(1+1)$ EA follows directly from results due to [6].

If the $(1+1)$ EA is changed in such a way that it only accepts bit strings with higher fitness the expected running time can be computed exactly. Because now the expected running time of the $(1+1)$ EA is $n^k(n/(n-1))^{n-k}$, if the initial bit string has $k \geqslant 1$ zeros. As the initial bit string is chosen randomly, the expected running time is now

$$\begin{aligned}
&\sum_{k=1}^{n} \binom{n}{k} 2^{-n} n^k \left(\frac{n}{n-1}\right)^{n-k} \\
&\quad = 2^{-n}\left(\left(\sum_{k=0}^{n} \binom{n}{k} n^k \left(\frac{n}{n-1}\right)^{n-k}\right) - \left(\frac{n}{n-1}\right)^{n}\right) \\
&\quad = 2^{-n}\left(\left(n+\frac{n}{n-1}\right)^{n} - \left(\frac{n}{n-1}\right)^{n}\right) \geqslant \left(\frac{n}{2}\right)^{n} = e^{n\ln(n/2)}. \qquad \square
\end{aligned}$$

So we have proven that the strategy of accepting equal-valued bit strings can improve the order of growth of the expected running time. But do functions exist, where the expected running time of the $(1+1)$ EA increases, when equal-valued bit strings are accepted? There exists a function $f\colon \{0,1\}^4 \to \mathbb{R}$ such that the expected running time is larger than 188 if equal-valued bit strings are accepted and smaller than 183 if only improvements are accepted [13]. But it is an open question, if there are functions such that the order of growth of the expected running time increases, when accepting equal-valued bit strings, too.

## 7. Conclusion

We have presented several methods to analyze the $(1+1)$ EA. These methods yield results for the classes of linear functions, polynomials of degree 2, and unimodal functions, and they can be used to obtain a hierarchy result. The next step is to analyze Evolutionary Algorithms which allow populations of individuals and crossover.

## References

[1] D.H. Ackley, A Connectionist Machine for Genetic Hillclimbers, Kluwer Academic Publishers, Boston, 1987.

[2] Th. Bäck, Optimal mutation rates in genetic search, in: S. Forrest (Ed.), Proc. 5th Internat. Conf. on Genetic Algorithms ICGA, Morgan Kaufman, San Mateo, CA, 1993, pp. 2–8.

[3] S. Droste, Th. Jansen, I. Wegener, A rigorous complexity analysis of the (1+1) Evolutionary Algorithm for linear functions with Boolean inputs, in: Proc. IEEE Internat. Conf. on Evolutionary Computation ICEC '98, IEEE Press, Piscataway, NJ, 1998, pp. 499–504.
[4] W. Feller, An Introduction to Probability Theory and its Applications, Vol. II, Wiley, New York, 1971.
[5] D.B. Fogel, Evolutionary Computation: Toward a New Philosophy of Machine Intelligence, IEEE Press, Piscataway, NJ, 1995.
[6] J. Garnier, L. Kallel, M. Schoenauer, Rigorous hitting times for binary mutations, Evolutionary Comput. 7 (2) (1999) 167–203.
[7] D.E. Goldberg, Genetic Algorithms in Search, Optimization, and Machine Learning, Addison-Wesley, Reading, MA, 1989.
[8] J.H. Holland, Adaption in Natural and Artificial Systems, University of Michigan, Michigan, 1975.
[9] J. Horn, D.E. Goldberg, K. Deb, Long path problems, in: Y. Davidor, H.-P. Schwefel, R. Männer (Eds.), Parallel Problem Solving from Nature PPSN III, Lecture Notes in Computer Science, Vol. 866, Springer, Berlin, 1994, pp. 149–158.
[10] A. Juels, M. Wattenberg, Hillclimbing as a baseline Method for the evaluation of stochastic optimization Algorithms, in: D.S. Touretzky, et al. (Eds.), Advances in Neural Information Processing Systems 8, MIT Press, Cambridge, MA, 1995, pp. 430–436.
[11] S. Kirkpatrick, C.D. Gelatt, M.P. Vecchi, Optimization by simulated annealing, Science 220 (1983) 671–680.
[12] J.R. Koza, Genetic Programming: On the Programming of Computers by means of Natural Selection, MIT Press, Cambridge, MA, 1992.
[13] R. Menke, personal communication, 1998.
[14] R. Motwani, P. Raghavan, Randomized Algorithms, Cambridge University Press, Cambridge, 1995.
[15] H. Mühlenbein, How Genetic Algorithms really work. Mutation and hill-climbing, in: R. Männer, R. Manderick (Eds.), Parallel Problem Solving from Nature PPSN II, North-Holland, Amsterdam, 1992, pp. 15–25.
[16] I. Rechenberg, Evolutionsstrategie '94, Frommann-Holzboog, Stuttgart, 1994.
[17] G. Rudolph, Convergence properties of evolutionary algorithms, Ph.D. Thesis, Verlag Dr. Kovač, Hamburg, 1997.
[18] G. Rudolph, How mutation and selection solve long-path problems in polynomial expected time, Evolutionary Comput. 4 (2) (1997) 195–205.
[19] H.-P. Schwefel, Evolution and Optimum Seeking, Wiley, New York, 1995.