Optimizing Linear Functions with Randomized Search Heuristics - The Robustness of Mutation

Witt, Carsten

Published in:
29th International Symposium on Theoretical Aspects of Computer Science (STACS 2012)

Link to article, DOI:
10.4230/LIPIcs.STACS.2012.420

Publication date:
2012

Document Version
Publisher's PDF, also known as Version of record

Optimizing Linear Functions with Randomized Search Heuristics – The Robustness of Mutation

Carsten Witt[1]

1 DTU Informatics, Technical University of Denmark

Abstract

The analysis of randomized search heuristics on classes of functions is fundamental for the understanding of the underlying stochastic process and the development of suitable proof techniques. Recently, remarkable progress has been made in bounding the expected optimization time of the simple (1+1) EA on the class of linear functions. We improve the best known bound in this setting from $(1.39 + o(1))en \ln n$ to $en \ln n + O(n)$ in expectation and with high probability, which is tight up to lower-order terms. Moreover, upper and lower bounds for arbitrary mutations probabilities p are derived, which imply expected polynomial optimization time as long as $p = O((\ln n)/n)$ and which are tight if $p = c/n$ for a constant c. As a consequence, the standard mutation probability $p = 1/n$ is optimal for all linear functions, and the (1+1) EA is found to be an optimal mutation-based algorithm. Furthermore, the algorithm turns out to be surprisingly robust since large neighborhood explored by the mutation operator does not disrupt the search.

1998 ACM Subject Classification F.2 [Analysis of algorithms and problem complexity]

Keywords and phrases Randomized Search Heuristics, Evolutionary Algorithms, Linear Functions, Running Time Analysis

Digital Object Identifier 10.4230/LIPIcs.STACS.2012.420

1 Introduction

Consider the following modified Coupon Collector process. The n bins, initially all empty, have weights. At each time step, go through the bins and flip the state (full/empty) of each bin independently with probability $1/n$. Then check whether the total weight of the full bins has decreased compared to the previous time step. If so, restore the previous configuration, otherwise keep the new one. How long does it take until all bins are full at the same time?

If all bins weigh the same, then an $O(n \log n)$ bound on the expected time follows along the famous analysis of the Coupon Collector Problem. However, if the weights are different, then the analysis becomes much more involved. In fact, this problem has been studied for more than a decade in the analysis of randomized search heuristics (RSH) and is known as the linear function problem there.

RSHs are general problem solvers that may be used when no problem-specific algorithm is available. Famous examples are simulated annealing, evolutionary computation, tabu search etc. In order to understand the working principles of RSHs, and to give theoretically founded advice on the applicability of certain RSHs, rigorous analyses of the runtime of RSHs have been conducted. This is a growing research area where many results have been obtained in recent years. It started off in the early 1990's [15] with the consideration of very simple evolutionary algorithms such as the well-known (1+1) EA on very simple example functions such as the well-known OneMax function. Later on, results regarding the runtime on classes of functions were derived [9, 11, 19, 20, e. g.] and important tools for the analysis

29th Symposium on Theoretical Aspects of Computer Science (STACS'12).
Editors: Christoph Dürr, Thomas Wilke; pp. 420–431

Leibniz International Proceedings in Informatics
Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl Publishing, Germany

were developed. Nowadays the state of the art in the field allows for the analysis of different types of search heuristics on problems from combinatorial optimization [16].

Recently, the analysis of evolutionary algorithms on linear functions has experienced a great renaissance. The first proof that the (1+1) EA optimizes any linear function in expected time $O(n \log n)$ by Droste et al. [9] was highly technical since it did not yet explicitly use the analytic framework of drift analysis [10], which allowed for a considerably simplified proof of the $O(n \log n)$ bound, see He and Yao [12] for the first complete proof using the method.[1] Another major improvement was made by Jägersküpper [13], who for the first time stated bounds on the implicit constant hidden in the $O(n \log n)$ term. This constant was finally improved by Doerr et al. [6] to the bound $(1.39 + o(1))en \ln n$ using a clean framework for the analysis of multiplicative drift [7]. The best known lower bound for general linear functions with non-zero weights is $en \ln n - O(n)$ and was also proven by Doerr et al. [6], building upon the OneMax function analyzed by Doerr et al. [3, 4].

The standard (1+1) EA flips each bit with probability $p = 1/n$ but also different values for the mutation probability p have been studied in the literature. Recently, it has been proved by Doerr and Goldberg [5] that the $O(n \log n)$ bound on the expected optimization time of the (1+1) EA still holds (also with high probability) if $p = c/n$ for an arbitrary constant c. This result uses the multiplicative drift framework mentioned above and a drift function being cleverly tailored towards the particular linear function. However, the analysis is also highly technical and does not yield explicit constants in the O-term. For $p = \omega(1/n)$, no runtime analyses were known so far.

In this paper, we prove that the (1+1) EA optimizes all linear functions in expected time $en \ln n + O(n)$, thereby closing the gap between the upper and the lower bound up to terms of lower order. Moreover, we show a general upper bound depending on the mutation probability p, which implies that the expected optimization time is polynomial as long as $p = O((\ln n)/n)$ (and $p = \Omega(1/\text{poly}(n))$). Since the expected optimization time is proved to be superpolynomial for $p = \omega((\ln n)/n)$, this implies a phase transition in the regime $\Theta((\ln n)/n)$. If the mutation probability is c/n for some constant c, the expected optimization time is proved to be $(1 \pm o(1))\frac{e^c}{c} n \ln n$. Altogether, we obtain that the standard choice $p = 1/n$ of the mutation probability is optimal for all linear functions. In fact, the lower bounds turn out to hold for the large class of so-called mutation-based EAs, in which the (1+1) EA with $p = 1/n$ is found to be an optimal algorithm.

Our findings are interesting both from a theoretical and practical perspective. On the theoretical side, it is noteworthy that $\frac{e^c}{c}$ is basically the expected waiting time for a mutation step that changes only a single bit. Hence, the mutation operator (in conjunction with the acceptance criterion) is surprisingly robust in the sense that steps flipping many bits do neither help nor harm. On the practical side, the optimality of $p = 1/n$ is remarkable since this seems to be the choice that is most often recommended by researchers in evolutionary computation [2]. Furthermore, the fact that the (1+1) EA is an optimal mutation-based algorithm emphasizes that it reflects the working principles of more complex EAs and that its runtime analysis can be crucial for obtaining results for more complex approaches.

The proofs of the upper bounds use the recent multiplicative drift theorem and a drift function adapted towards both the linear function and the mutation probability. As a consequence from our main result, we obtain the results by Doerr and Goldberg [5] with less effort and explicit constants in front of the $n \ln n$-term. All these bounds hold also with

[1] Note, however, that not the original (1+1) EA but a variant rejecting offspring of equal fitness is studied in that paper.

high probability, which follows from the recent tail bounds added to the multiplicative drift theorem by Doerr and Goldberg [5]. The lower bounds are based on a new multiplicative drift theorem for lower bounds. By deriving very exact results, we show that the research area is maturing and provides for very strong and, at the same time, general tools.

This paper is structured as follows. Section 2 sets up definitions, notations and other preliminaries. Section 3 summarizes and explains the main results. In Sections 4 and 5, respectively, we prove an upper bound for general mutation probabilities and a refined result for $p = 1/n$. Lower bounds are shown in Section 6. We finish with some conclusions. Due to space limitations, several proofs had to be omitted from this paper.

2 Preliminaries

The (1+1) EA is a basic search heuristic for the optimization of pseudo-boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$. It reflects the typical behavior of more complicated evolutionary algorithms, serves as basis for the study of more complex approaches and is therefore intensively investigated in the theory of RSHs [1]. For the case of minimization, it is defined as Algorithm 1.

Algorithm 1 (1+1) EA

$t := 0$.
choose uniformly at random an initial bit string $x_0 \in \{0,1\}^n$.
repeat
  create x' by flipping each bit in x_t independently with prob. $p \leq 1/2$ *(mutation)*.
  $x_{t+1} := x'$ if $f(x') \leq f(x_t)$, and $x_{t+1} := x_t$ otherwise *(selection)*.
  $t := t + 1$.
until forever.

The (1+1) EA can be considered a simple hill-climber where search points are drawn from a stochastic neighborhood based on the mutation operator. The parameter p, where $0 < p \leq 1/2$, is often chosen as $1/n$, which then is called *standard mutation probability*. We call a mutation from x_t to x' *accepted* if $f(x') \leq f(x_t)$, i. e., if the new search point is taken over; otherwise we call it *rejected*. In our theoretical studies, we ignore the fact that the algorithm in practice will be stopped at some time. The *runtime* (synonymously, *optimization time*) of the (1+1) EA is defined as the first random point in time t such that the search point x_t has optimal, i. e., minimum f-value. This corresponds to the number of f-evaluations until reaching the optimum. In many cases, one is aiming for results on the expected optimization time. Here, we also prove results that hold with high probability, which means probability $1 - o(1)$.

The (1+1) EA is also an instantiation of the algorithmic scheme that is called *mutation-based EA* by Sudholt [17] and is displayed as Algorithm 2. It is a general population-based approach that includes many variants of evolutionary algorithms with parent and offspring populations as well as parallel evolutionary algorithms. Any mechanism for managing the populations, which are multisets, is allowed as long as the mutation operator is the only variation operator and follows the independent bit-flip property with probability $0 < p \leq 1/2$. Again the smallest t such that x_t is optimal defines the runtime. Sudholt has proved for $p = 1/n$ that no mutation-based EA can locate a unique optimum faster than the (1+1) EA can optimize OneMax. We will see that the (1+1) EA is the best mutation-based EA on a broad class of functions, also for different mutation probabilities.

Algorithm 2 Scheme of a mutation-based EA

for $t := 0 \to \mu - 1$ **do**
  choose $x_t \in \{0,1\}^n$ uniformly at random.
end for
repeat
  select a parent $x \in \{x_0, \dots, x_t\}$ according to t and $f(x_0), \dots, f(x_t)$.
  create x_{t+1} by flipping each bit in x independently with probability $p \leq 1/2$.
  $t := t + 1$.
until forever.

Throughout this paper, we deal with linear functions. A function $f\colon \{0,1\}^n \to \mathbb{R}$ is called *linear* if it can be written as $f(x_n, \dots, x_1) = w_n x_n + \cdots + w_1 x_1 + w_0$. As common in the analysis of the (1+1) EA, we assume w. l. o. g. that $w_0 = 0$ and $w_n \geq \cdots \geq w_1 > 0$ hold. Search points are read from x_n down to x_1 such that x_n, the most significant bit, is said to be on the left-hand side and x_1, the least significant bit, on the right-hand side. Since it fits the proof techniques more naturally, we assume also w. l. o. g. that the (1+1) EA (or, more generally, the mutation-based EA at hand) is minimizing f, implying that the all-zeros string is the optimum. Our assumptions do not lose generality since we can permute bits and negate the weights of a linear function without affecting the stochastic behavior of the (1+1) EA/mutation-based EA.

The probably best-studied linear function is $\textsc{OneMax}(x_n, \dots, x_1) = x_n + \cdots + x_1$, occasionally also called the *CountingOnes* problem (which would be the more appropriate name here since we will be minimizing the function). In this paper, we will see that on the one hand, OneMax is not only the easiest linear function definition-wise but also in terms of expected optimization time. On the other hand, the upper bounds obtained for OneMax hold for every linear function up to lower-order terms. Hence, surprisingly the (1+1) EA is basically as efficient on an arbitrary linear function as it is on OneMax. This underlines the robustness of the randomized search heuristic and, in retrospect and for the future, is a strong motivation to investigate the behavior of RSHs on the OneMax problem thoroughly.

Our proofs of the forthcoming upper bounds use the multiplicative drift theorem in its most recent version (cf. [5] and [7]). The key idea of multiplicative drift is to identify a time-independent relative progress called drift.

▶ **Theorem 1** (Multiplicative Drift, Upper Bound). *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum 1. Let $\{X^{(t)}\}_{t \geq 0}$ be a sequence of random variables over $S \cup \{0\}$. Let T be the random first point in time $t \geq 0$ for which $X^{(t)} = 0$.*

Suppose that there exists a $\delta > 0$ such that

$$E(X^{(t)} - X^{(t+1)} \mid X^{(t)} = s) \;\geq\; \delta s$$

for all $s \in S$ with $\text{Prob}(X^{(t)} = s) > 0$. Then for all $s_0 \in S$ with $\text{Prob}(X^{(0)} = s_0) > 0$,

$$E(T \mid X^{(0)} = s_0) \;\leq\; \frac{\ln(s_0) + 1}{\delta}.$$

Moreover, it holds that $\text{Prob}(T > (\ln(s_0) + t)/\delta)) \leq e^{-t}$.

As an easy example application, consider the (1+1) EA on OneMax and let $X^{(t)}$ denote the number of one-bits at time t. As worse search points are not accepted, $X^{(t)}$ is non-increasing over time. We obtain $E(X^{(t)} - X^{(t+1)} \mid X^{(t)} = s) \geq s(1/n)(1-1/n)^{n-1} \geq s/(en)$,

in other words a multiplicative drift of at least $\delta = 1/(en)$, since there are s disjoint single-bit flips that decrease the X-value by 1. Theorem 1 applied with $\delta = 1/(en)$ and $\ln(X^{(0)}) \leq \ln n$ gives us the upper bound $en(\ln n + 1)$ on the expected optimization time, which is basically the same as the classical method of fitness-based partitions [17, 18] or coupon collector arguments [14] would yield.

On a general linear function, it is not necessarily a good choice to let $X^{(t)}$ count the current number of one-bits. Consider, e. g., the well-known function $\textsc{BinVal}(x_n, \dots, x_1) = \sum_{i=1}^{n} 2^{i-1} x_i$. The (1+1) EA might replace the search point $(1, 0, \dots, 0)$ by the better search point $(0, 1, \dots, 1)$, amounting to a loss of $n-2$ zero-bits. More generally, replacing $(1, 0, \dots, 0)$ by a better search point is equivalent to flipping the leftmost one-bit. In such a step, an expected number of $(n-1)p$ zero-bits flip, which decreases the expected number of zero-bits by only $1-(n-1)p$. The latter expectation (the so-called additive drift) is only $1/n$ for the standard mutation probability $p = 1/n$ and might be negative for larger p. Therefore, $X^{(t)}$ is typically defined as $X^{(t)} := g(x^{(t)})$, where $x^{(t)}$ is the current search point at time t and $g(x_n, \dots, x_1)$ is another linear function called *drift function* or *potential function*. Doerr et al. [7] use $x_1 + \dots + x_{n/2} + (5/4)(x_{n/2+1} + \dots + x_n)$ as potential function in their application of the multiplicative drift theorem. This leads to a good lower bound on the multiplicative drift on the one hand and a small maximum value of $X^{(t)}$ on the other hand. In our proofs of upper bounds in the Sections 4 and 5, it is crucial to define appropriate potential functions.

For the lower bounds in Section 6, we need the following variant of the multiplicative drift theorem.

▶ **Theorem 2** (Multiplicative Drift, Lower Bound). *Let $S \subseteq \mathbb{R}$ be a finite set of positive numbers with minimum 1. Let $\{X^{(t)}\}_{t \geq 0}$ be a sequence of random variables over S, where $X^{(t+1)} \leq X^{(t)}$ for any $t \geq 0$, and let $s_{\min} > 0$. Let T be the random first point in time $t \geq 0$ for which $X^{(t)} \leq s_{\min}$. If there exist positive reals $\beta, \delta \leq 1$ such that for all $s > s_{\min}$ and all $t \geq 0$ with $\operatorname{Prob}(X^{(t)} = s) > 0$ it holds that*

1. $E\big(X^{(t)} - X^{(t+1)} \mid X^{(t)} = s\big) \leq \delta s$,
2. $\operatorname{Prob}(X^{(t)} - X^{(t+1)} \geq \beta s \mid X^{(t)} = s) \leq \beta\delta / \ln s$,

then for all $s_0 \in S$ with $\operatorname{Prob}(X^{(0)} = s_0) > 0$,

$$E\Big(T \mid X^{(0)} = s_0\Big) \geq \frac{\ln(s_0) - \ln(s_{\min})}{\delta} \cdot \frac{1-\beta}{1+\beta}.$$

The lower-bound version includes a condition on the maximum stepwise progress and requires monotonicity since the upper bound can be very pessimistic otherwise. As a technical detail, we allows for a positive target $s_{\min}$, which is required in our applications.

3 Summary of Main Results

We now list the main consequences from the lower bounds and upper bounds that we will prove in the following sections.

▶ **Theorem 3.** *On any linear function, the following holds for the expected optimization time $E(T_p)$ of the (1+1) EA with mutation probability p.*

1. *If $p = \omega((\ln n)/n)$ or $p = o(1/\operatorname{poly}(n))$ then $E(T_p)$ is superpolynomial.*
2. *If $p = \Omega(1/\operatorname{poly}(n))$ and $p = O((\ln n)/n)$ then $E(T_p)$ is polynomial.*
3. *If $p = c/n$ for a constant c then $E(T_p) = (1 \pm o(1))\frac{e^c}{c} n \ln n$.*
4. *$E(T_p)$ is minimized for mutation probability $p = 1/n$ if n is large enough.*
5. *No mutation-based EA has an expected optimization time that is smaller than $E(T_{1/n})$ (up to lower-order terms).*

In fact, our forthcoming analyses are more precise; in particular, we do not state available tails on the upper bounds above and leave them in the more general, but also more complicated Theorem 4 in Section 4. The first statement of our summarizing Theorem 3 follows from the Theorems 10 and 11 in Section 6. The second statement is proven in Corollary 6, which follows from the already mentioned Theorem 4. The third statement takes together the Corollaries 5 and 12. Since e^c/c is minimized for $c = 1$, the fourth statement follows from the third one in conjunction with Corollary 12. The fifth statement is also contained in the Theorems 10 and 11.

It is worth noting that the optimality of $p = 1/n$ apparently was never proven rigorously before, not even for the case of OneMax[2], where tight upper and lower bounds on the expected optimization time were only available for the standard mutation probability [4, 17]. For the general case of linear functions, the strongest previous result said that $p = \Theta(1/n)$ is optimal [9]. Our result on the optimality of the mutation probability $1/n$ is interesting since this is the commonly recommended choice by practitioners.

4 Upper Bounds

In this section, we show a general upper bound that applies to any non-trivial mutation probability.

▶ **Theorem 4.** *On any linear function, the optimization time of the (1+1) EA with mutation probability $0 < p \leq 1/2$ is at most*

$$(1-p)^{1-n}\left(\frac{n\alpha^2(1-p)^{1-n}}{\alpha-1} + \frac{\alpha}{\alpha-1}\frac{\ln(1/p)+(n-1)\ln(1-p)+t}{p}\right) =: b(t)$$

with probability at least $1-e^{-t}$, and it is at most $b(1)$ in expectation, where $\alpha > 1$ can be chosen arbitrarily (also depending on n).

Before we prove the theorem, we note two important consequences in more readable form. The first one (Corollary 5) displays upper bounds for mutation probabilities c/n. The second one (Corollary 6) is used in Theorem 3 above, which states a phase transition from polynomial to superpolynomial expected optimization times at mutation probability $p = \Theta((\ln n)/n)$.

▶ **Corollary 5.** *On any linear function, the optimization time of the (1+1) EA with mutation probability $p = c/n$, where $c > 0$ is a constant, is bounded from above by $(1+o(1))((e^c/c)n\ln n)$ with probability $1-o(1)$ and also in expectation.*

▶ **Corollary 6.** *On any linear function, the optimization time of the (1+1) EA with mutation probability $p = O((\ln n)/n)$ and $p = \Omega(1/\mathrm{poly}(n))$ is polynomial with probability $1-o(1)$ and also in expectation.*

The proof of Theorem 4 uses an adaptive potential function as in Doerr and Goldberg [5]. That is, the random variables $X^{(t)}$ used in Theorem 1 map the current search point of the (1+1) EA via a potential function to some value in a way that depends also on the linear function at hand. As a special case, if the given linear function happens to be OneMax, $X^{(t)}$ just counts the number of one-bits at time t. The general construction shares some

[2] However, a recent technical report extending Sudholt [17] shows the optimality of $p = 1/n$ in the case of OneMax using a different approach, see `http://arxiv.org/abs/1109.1504`.

similarities with the one in Doerr and Goldberg [5], but both construction and proof are significantly less involved. Moreover, we can also consider $p = \omega(1/n)$.

Proof of Theorem 4. Let $f(x) = w_n x_n + \cdots + w_1 x_1$ be the linear function at hand. Define

$$\gamma_i := \left(1 + \frac{\alpha p}{(1-p)^{n-1}}\right)^{i-1}$$

for $1 \le i \le n$, and let $g(x) = g_n x_n + \cdots + g_1 x_1$ be the potential function defined by $g_1 := 1 = \gamma_1$ and

$$g_i \;:=\; \min\left\{\gamma_i,\, g_{i-1} \cdot \frac{w_i}{w_{i-1}}\right\}$$

for $2 \le i \le n$. Note that the g_i are non-decreasing w. r. t. i. Intuitively, if the ratio of w_i and w_{i-1} is too extreme, the minimum function caps it appropriately, otherwise g_i and g_{i-1} are in the same ratio. We consider the stochastic process $X^{(t)} := g(a^{(t)})$, where $a^{(t)}$ is the current search point of the (1+1) EA at time t. Obviously, $X^{(t)} = 0$ if and only if f has been optimized.

Let $\Delta_t := X^{(t)} - X^{(t+1)}$. We will show below that

$$E(\Delta_t \mid X^{(t)} = s) \;\ge\; s \cdot p \cdot (1-p)^{n-1} \cdot \left(1 - \frac{1}{\alpha}\right). \qquad (*)$$

The initial value satisfies

$$X^{(0)} \;\le\; g_n + \cdots + g_1 \;\le\; \sum_{i=1}^{n} \gamma^i \;\le\; \frac{\left(1 + \frac{\alpha p}{(1-p)^{n-1}}\right)^n - 1}{\alpha p (1-p)^{1-n}} \;\le\; \frac{e^{n\alpha p (1-p)^{1-n}}}{\alpha p (1-p)^{1-n}},$$

which means

$$\ln(X^{(0)}) \;\le\; n\alpha p (1-p)^{1-n} + \ln(1/p) + \ln((1-p)^{n-1}).$$

The multiplicative drift theorem (Theorem 1) yields that the optimization time T is bounded from above by

$$\frac{\ln(X_0) + t}{p(1-p)^{n-1}(1-1/\alpha)} \;\le\; \frac{\alpha\left(n\alpha p(1-p)^{1-n} + \ln(1/p) + \ln((1-p)^{n-1}) + t\right)}{(\alpha-1)p(1-p)^{n-1}} \;=\; b(t)$$

with probability at least $1 - e^{-t}$, and $E(T) = b(1)$, which proves the theorem.

To show $(*)$, we fix an arbitrary current value s and an arbitrary search point $a^{(t)}$ satisfying $g(a^{(t)}) = s$. In the following, we implicitly assume $X^{(t)} = s$ but mostly omit this for the sake of readability. We denote by $I := \{i \mid a_i^{(t)} = 1\}$ the index set of the one-bits in $a^{(t)}$ and by $Z := \{1, \ldots, n\} \setminus I$ the zero-bits. We assume $I \neq \emptyset$ since there is nothing to show otherwise. Denote by a' the random (not necessarily accepted) offspring produced by the (1+1) EA when mutating $a^{(t)}$ and by $a^{(t+1)}$ the next search point after selection. Recall that $a^{(t+1)} = a'$ if and only if $f(a') \le f(a^{(t)})$. In the following, we will use the event A that $a^{(t+1)} = a' \neq a^{(t)}$ since obviously $\Delta_t = 0$ otherwise. Let $I^* := \{i \in I \mid a'_i = 0\}$ be the random set of flipped one-bits and $Z^* := \{i \in Z \mid a'_i = 1\}$ be the set of flipped zero-bits in a' (not conditioned on A). Note that $I^* \neq \emptyset$ if A occurs.

We need further definitions to analyze the drift carefully. For $i \in I$, we define $k(i) := \max\{j \le i \mid g_j = \gamma_j\}$ as the most significant position to the right of i (possibly i itself) where the potential function might be capping; note that $k(i) \ge 1$ since $g_1 = \gamma_1$. Let

$L(i) := \{k(i), \dots, n\} \cap Z$ be the set of zero-bits left of (and including) $k(i)$ and let $R(i) := \{1, \dots, k(i)-1\} \cap Z$ be the remaining zero-bits. Both sets may be empty. For event A to occur, it is necessary that there is some $i \in I$ such that bit i flips to zero and

$$\sum_{j \in I^*} w_j - \sum_{j \in Z^* \cap L(i)} w_j \geq 0$$

since we are taking only zero-bits out of consideration. Now, for $i \in I$, let A_i be the event that

1. i is the leftmost flipping one-bit (i. e., $i \in I^*$ and $\{i+1, \dots, n\} \cap I^* = \emptyset$) and
2. $\sum_{j \in I^*} w_j - \sum_{j \in Z^* \cap L(i)} w_j \geq 0$.

If none of the A_i occurs, $\Delta_t = 0$. Furthermore, the A_i are mutually disjoint.

For any $i \in I$, Δ_t can be written as the sum of the two terms

$$\Delta_L(i) := \sum_{j \in I^*} g_j - \sum_{j \in Z^* \cap L(i)} g_j \quad \text{and} \quad \Delta_R(i) := - \sum_{j \in Z^* \cap R(i)} g_j.$$

By the law of total probability and the linearity of expectation, we have

$$E(\Delta_t) = \sum_{i \in I} E(\Delta_L(i) \mid A_i) \cdot \text{Prob}(A_i) + E(\Delta_R(i) \mid A_i) \cdot \text{Prob}(A_i). \qquad (**)$$

In the following, the bits in $R(i)$ are pessimistically assumed to flip to 1 independently with probability p each if A_i happens. This leads to $E(\Delta_R(i) \mid A_i) \geq -p \sum_{j \in R(i)} g_j$.

In order to estimate $E(\Delta_L(i))$, we carefully inspect the relation between the weights of the original function and the potential function. By definition, we obtain $g_j/g_{k(i)} = w_j/w_{k(i)}$ for $k(i) \leq j \leq i$ and $g_j/g_{k(i)} \leq w_j/w_{k(i)}$ for $j > i$ whereas $g_j/g_{k(i)} \geq w_j/w_{k(i)}$ for $j < k(i)$. Hence, if A_i occurs then $g_j \geq g_{k(i)} \cdot \frac{w_j}{w_{k(i)}}$ for $j \in I^*$ (since i is the leftmost flipping one-bit) whereas $g_j \leq g_{k(i)} \cdot \frac{w_j}{w_{k(i)}}$ for $j \in L(i)$. Together, we obtain under $A(i)$ the nonnegativity of the random variable $\Delta_L(i)$:

$$\begin{aligned} \Delta_L(i) \mid A_i &= \sum_{j \in I^* \mid A_i} g_j - \sum_{j \in (Z^* \cap L(i)) \mid A_i} g_j \\ &\geq \sum_{j \in I^* \mid A_i} g_{k(i)} \cdot \frac{w_j}{w_{k(i)}} - \sum_{j \in (Z^* \cap L(i)) \mid A_i} g_{k(i)} \cdot \frac{w_j}{w_{k(i)}} \geq 0 \end{aligned}$$

using the definition of A_i.

Now let $S_i := \{|Z^* \cap L(i)| = 0\}$ be the event that no zero-bit from $L(i)$ flips. Using the law of total probability, we obtain that

$$\begin{aligned} E(\Delta_L(i) \mid A_i) \cdot \text{Prob}(A_i) &= E(\Delta_L(i) \mid A_i \cap S_i) \cdot \text{Prob}(A_i \cap S_i) \\ &\quad + E(\Delta_L(i) \mid A_i \cap \overline{S_i}) \cdot \text{Prob}(A_i \cap \overline{S_i}). \end{aligned}$$

Since $\Delta_L(i) \mid A_i \geq 0$, the conditional expectations are non-negative. We bound the second term on the right-hand side by 0. In conjunction with $(**)$, we get

$$E(\Delta_t) \geq \sum_{i \in I} E(\Delta_L(i) \mid A_i \cap S_i) \cdot \text{Prob}(A_i \cap S_i) + E(\Delta_R(i) \mid A_i) \cdot \text{Prob}(A_i).$$

Obviously, $E(\Delta_L(i) \mid A_i \cap S_i) \geq g_i$. We estimate $\text{Prob}(A_i \cap S_i) \geq p(1-p)^{n-1}$ since it is sufficient to flip only bit i and $\text{Prob}(A_i) \leq p$ since it is necessary to flip this bit. Further

above, we have bounded $E(\Delta_R(i) \mid A_i)$. Taking everything together, we get

$$E(\Delta_t) \geq \sum_{i \in I} \left(p(1-p)^{n-1} g_i - p^2 \sum_{j \in R(i)} g_j \right)$$
$$\geq \sum_{i \in I} \left(p(1-p)^{n-1} \frac{g_i}{g_{k(i)}} \gamma_{k(i)} - p^2 \sum_{j=1}^{k(i)-1} \gamma_j \right).$$

The term for i equals

$$p(1-p)^{n-1} \frac{g_i}{g_{k(i)}} \left(1 + \frac{\alpha p}{(1-p)^{n-1}}\right)^{k(i)-1} - \frac{p^2 \cdot \left(\left(1 + \frac{\alpha p}{(1-p)^{n-1}}\right)^{k(i)-1} - 1 \right)}{\left(\frac{\alpha p}{(1-p)^{n-1}}\right)}$$
$$\geq \left(1 - \frac{1}{\alpha}\right) p(1-p)^{n-1} \frac{g_i}{g_{k(i)}} \left(1 + \frac{\alpha p}{(1-p)^{n-1}}\right)^{k(i)-1} = \left(1 - \frac{1}{\alpha}\right) p(1-p)^{n-1} g_i,$$

where the inequality uses $g_i \geq g_{k(i)}$. Hence,

$$E(\Delta_t) \geq \sum_{i \in I} \left(1 - \frac{1}{\alpha}\right) p(1-p)^{n-1} g_i = \left(1 - \frac{1}{\alpha}\right) p(1-p)^{n-1} g(a^{(t)}),$$

which proves $(*)$, and, therefore, the theorem. □

5 Refined Upper Bound for Mutation Probability $1/n$

In this section, we consider the standard mutation probability $p = 1/n$ and refine the result from Corollary 5. More precisely, we obtain that the lower order-terms are $O(n)$. The proof is shorter and uses a simpler potential function.

▶ **Theorem 7.** *On any linear function, the expected optimization time of the (1+1) EA with $p = 1/n$ is at most $en \ln n + 2en + O(1)$, and the probability that the optimization time exceeds $en \ln n + (1+t)en + O(1)$ is at most e^{-t}.*

6 Lower Bounds

In this section, we state lower bounds that prove the results from Theorem 4 to be tight up to lower-order terms for a wide range of mutation probabilities. Moreover, we show that the lower bounds hold for the very large class of mutation-based algorithms (Algorithm 2). Recall that a list of the most important consequences is given above in Theorem 3. For technical reasons, we split the proof of the lower bounds into two main cases, namely $p = O(n^{-2/3-\varepsilon})$ and $p = \Omega(n^{\varepsilon-1})$ for any constant $\varepsilon > 0$. The proofs go back to OneMax as a worst case, as outlined in the following subsection.

6.1 OneMax as Easiest Linear Function

Doerr et al. [6] show with respect to the (1+1) EA with standard mutation probability $1/n$ that OneMax is the "easiest" function from the class of functions with unique global optimum, which comprises the class of linear functions. More precisely, the expected optimization time on OneMax is proved to be smallest within the class.

We will generalize this result to $p \leq 1/2$ with moderate additional effort. In fact, we will relate the behavior of an arbitrary mutation-based EA on OneMax to the (1+1) EA_μ in a similar way to Sudholt [17, Section 7]. The latter algorithm, displayed as Algorithm 3, creates search points uniformly at random from time 0 to time $\mu - 1$ and then chooses a best one from these to be the current search point at time $\mu - 1$; afterwards it works as the standard (1+1) EA. Note that we obtain the standard (1+1) EA for $\mu = 1$. Moreover, we will only consider the case $\mu = \text{poly}(n)$ in order to bound the running time of the initialization. This makes sense since a unique optimum (such as the all-zeros string for OneMax) is with overwhelming probability not found even when drawing $2^{\sqrt{n}}$ random search points.

Algorithm 3 (1+1) EA_μ

for $t := 0 \to \mu - 1$ **do**
  choose $x_t \in \{0,1\}^n$ uniformly at random.
end for
$x_t := \arg\min\{f(x) \mid x \in \{x_0, \ldots, x_t\}\}$ (breaking ties uniformly).
repeat
  create x' by flipping each bit in x_t independently with prob. p.
  $x_{t+1} := x'$ if $f(x') \leq f(x_t)$, and $x_{t+1} := x_t$ otherwise.
  $t := t + 1$.
until forever.

Our analyses need the monotonicity statement from Lemma 8 below, which is similar to Lemma 11 in Doerr et al. [6] and whose proof is already sketched in Droste et al. [8, Section 5]. Note, however, that Doerr et al. [6] only consider $p = 1/n$ and have a stronger statement for this case. More precisely, they show $\text{Prob}(|mut(a)|_1 = j) \geq \text{Prob}(|mut(b)|_1 = j)$, which does not hold for large p. Here and hereinafter, $|x|_1$ denotes the number of ones in a bit string x.

▶ **Lemma 8.** *Let $a, b \in \{0,1\}^n$ be two search points satisfying $|a|_1 < |b|_1$. Denote by $mut(x)$ the random string obtained by mutating each bit of x independently with probability p. Let $0 \leq j \leq n$ be arbitrary. If $p \leq 1/2$ then $\text{Prob}(|mut(a)|_1 \leq j) \geq \text{Prob}(|mut(b)|_1 \leq j)$.*

The following theorem is a generalization of Theorem 9 by Doerr et al. [6] to the case $p \leq 1/2$ instead of $p = 1/n$. However, we not only generalize to higher mutation probabilities, but also also consider the more general class of mutation-based algorithms. Finally, we prove stochastic ordering, while Doerr et al. [6] inspect only the expected optimization times. Still, many ideas of the original proof can be taken over and be combined with the proof of Theorem 5 in Sudholt [17].

▶ **Theorem 9.** *Consider a mutation-based EA A with population size μ and mutation probability $p \leq 1/2$ on any function with unique global optimum. Then the optimization time of A is stochastically at least as large as the optimization time of the (1+1) EA_μ on* OneMax.

6.2 Large Mutation Probabilities

It is not too difficult to show that mutation probabilities $p = \Omega(n^{\varepsilon-1})$, where $\varepsilon > 0$ is an arbitrary constant, make the (1+1) EA (and also the (1+1) EA_μ) flip too many bits for it to optimize linear functions efficiently.

▶ **Theorem 10.** *On any linear function, the optimization time of an arbitrary mutation-based EA with $\mu = \text{poly}(n)$ and $p = \Omega(n^{\varepsilon-1})$ for some constant $\varepsilon > 0$, is bounded from below by $2^{\Omega(n^\varepsilon)}$ with probability $1 - 2^{-\Omega(n^\varepsilon)}$.*

6.3 Small Mutation Probabilities

We now turn to mutation probabilities that are bounded from above by roughly $1/n^{2/3}$. Here quite precise lower bounds can be obtained.

▶ **Theorem 11.** *On any linear function, the expected optimization time of an arbitrary mutation-based EA with $\mu = \text{poly}(n)$ and $p = O(n^{-2/3-\varepsilon})$ is bounded from below by $(1 - o(1)) \cdot (1-p)^{-n}(1/p)\min\{\ln n, \ln(1/(p^3n^2))\}$.*

As a consequence from Theorem 11, we obtain that the bound from Theorem 4 is tight (up to lower-order terms) for the (1+1) EA as long as $\ln(1/(p^3n^2)) = \ln n - o(\ln n)$. This condition is weaker than $p = O((\ln n)/n)$. If $p = \omega((\ln n)/n)$ or $p = o(1/\text{poly}(n))$, then Theorem 11 in conjunction with Theorem 10 imply superpolynomial expected optimization time. Thus, the bounds are tight for all p that allow polynomial optimization times.

We state another important consequence, implying the statement from Theorem 3 that using the (1+1) EA with mutation probability $1/n$ is optimal for any linear function.

▶ **Corollary 12.** *On any linear function, the expected optimization time of a mutation-based EA with $\mu = \text{poly}(n)$ and $p = c/n$, where $c > 0$ is a constant, is bounded from below by $(1 - o(1))((e^c/c)n \ln n)$. If $p = \omega(1/n)$ or $p = o(1/n)$, the expected optimization time is $\omega(n \ln n)$.*

Finally, we remark that the expected optimization time of the (1+1) EA with $p = 1/n$ on OneMax is known to be $en \ln n - \Theta(n)$ [4]. Hence, in conjunction with the Theorems 7 and 9, we obtain for $p = 1/n$ that the expected optimization time of the (1+1) EA varies by at most an additive term $\Theta(n)$ within the class of linear functions.

Conclusions

We have presented new bounds on the expected optimization time of the simple (1+1) EA on the class of linear functions. The results are now tight up to lower-order terms, which applies to any mutation probability $p = O((\ln n)/n)$. This means that $1/n$ is the optimal mutation probability on any linear function. We have for the first time studied the case $p = \omega(1/n)$ and proved a phase transition from polynomial to exponential running time in the regime $\Theta((\ln n)/n)$. The lower bounds show that OneMax is the easiest linear function, and they apply not only to the (1+1) EA but also to the large class of mutation-based EAs. They so exhibit the (1+1) EA as optimal mutation-based algorithm on linear functions. The upper bounds hold with high probability. The analyses shed light on the working principles of randomized search heuristics on simple problems and prove that they can be surprisingly robust with respect to their parametrization. As proof techniques, we have used and further developed multiplicative drift analysis in conjunction with adaptive potential functions. In the future, we are confident to see these techniques applied to the analysis of other RSHs.

Acknowledgments

The author thanks Benjamin Doerr, Timo Kötzing, Per Kristian Lehre, Dirk Sudholt and Carola Winzen for insightful discussions and useful suggestions. Moreover, he thanks Daniel Johannsen for pointing out a simplification of the proof of Theorem 4.

References

1 Anne Auger and Benjamin Doerr. *Theory of Randomized Search Heuristics – Foundations and Recent Developments*. World Scientific Publishing, 2011.

2 Thomas Bäck. Optimal mutation rates in genetic search. In *Proc. of ICGA '93*, pages 2–8. Morgan Kaufmann, 1993.

3 Benjamin Doerr, Mahmoud Fouz, and Carsten Witt. Quasirandom evolutionary algorithms. In *Proc. of GECCO '10*, pages 1457–1464. ACM Press, 2010.

4 Benjamin Doerr, Mahmoud Fouz, and Carsten Witt. Sharp bounds by probability-generating functions and variable drift. In *Proc. of GECCO '11*, pages 2083–2090. ACM Press, 2011.

5 Benjamin Doerr and Leslie Ann Goldberg. Adaptive drift analysis. *Algorithmica*, 2012. To appear; preprint: `http://arxiv.org/abs/1108.0295`.

6 Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Drift analysis and linear functions revisited. In *Proc. of CEC '10*, pages 1–8. IEEE Press, 2010.

7 Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. In *Proc. of GECCO '10*, pages 1449–1456. ACM Press, 2010.

8 Stefan Droste, Thomas Jansen, and Ingo Wegener. A natural and simple funtions which is hard for all evolutionary algorithms. In *Proc. of IECON '00*, pages 2704–2709, 2000. DOI: 10.1109/IECON.2000.972425.

9 Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

10 Bruce Hajek. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability*, 13(3):502–525, 1982.

11 Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:57–85, 2001.

12 Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

13 Jens Jägersküpper. A blend of markov-chain and drift analysis. In *Proc. of PPSN '08*, volume 5199 of *LNCS*, pages 41–51. Springer, 2008.

14 Rajeev Motwani and Prabhakar Raghavan. *Randomized Algorithms*. Cambridge University Press, 1995.

15 Heinz Mühlenbein. How genetic algorithms really work: I. Mutation and hillclimbing. In *Proc. of PPSN '92*, pages 15–26. Elsevier, 1992.

16 Frank Neumann and Carsten Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Natural Computing Series. Springer, 2010.

17 Dirk Sudholt. General lower bounds for the running time of evolutionary algorithms. In *Proc. of PPSN '10*, volume 6238 of *LNCS*, pages 124–133. Springer, 2010. Extended version: `http://arxiv.org/abs/1109.1504`.

18 Ingo Wegener. Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In Ruhul Sarker, Masoud Mohammadian, and Xin Yao, editors, *Evolutionary Optimization*. Kluwer Academic Publishers, 2001.

19 Ingo Wegener and Carsten Witt. On the analysis of a simple evolutionary algorithm on quadratic pseudo-boolean functions. *Journal of Discrete Algorithms*, 3(1):61–78, 2005.

20 Ingo Wegener and Carsten Witt. On the optimization of monotone polynomials by simple randomized search heuristics. *Combinatorics, Probability & Computing*, 14(1-2):225–247, 2005.