



HAL
open science

A Tight Runtime Analysis for the $(\mu + \lambda)$ EA

Denis Antipov, Benjamin Doerr

► **To cite this version:**

Denis Antipov, Benjamin Doerr. A Tight Runtime Analysis for the $(\mu + \lambda)$ EA. *Algorithmica*, 2021, 83 (4), pp.1054-1095. 10.1007/s00453-020-00731-5 . hal-03353025

HAL Id: hal-03353025

<https://hal.science/hal-03353025>

Submitted on 23 Sep 2021

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

A Tight Runtime Analysis for the $(\mu + \lambda)$ EA*

Denis Antipov
ITMO University
Saint-Petersburg
Russia

Benjamin Doerr
Laboratoire d'Informatique (LIX)
CNRS, École Polytechnique
Institut Polytechnique de Paris
Palaiseau, France

January 22, 2020

Abstract

Despite significant progress in the theory of evolutionary algorithms, the theoretical understanding of evolutionary algorithms which use non-trivial populations remains challenging and only few rigorous results exist. Already for the most basic problem, the determination of the asymptotic runtime of the $(\mu + \lambda)$ evolutionary algorithm on the simple ONEMAX benchmark function, only the special cases $\mu = 1$ and $\lambda = 1$ have been solved.

In this work, we analyze this long-standing problem and show the asymptotically tight result that the runtime T , the number of iterations until the optimum is found, satisfies

$$E[T] = \Theta\left(\frac{n \log n}{\lambda} + \frac{n}{\lambda/\mu} + \frac{n \log^+ \log^+(\lambda/\mu)}{\log^+(\lambda/\mu)}\right),$$

where $\log^+ x := \max\{1, \log x\}$ for all $x > 0$.

The same methods allow to improve the previous-best $O(\frac{n \log n}{\lambda} + n \log \lambda)$ runtime guarantee for the $(\lambda + \lambda)$ EA with fair parent selection to a tight $\Theta(\frac{n \log n}{\lambda} + n)$ runtime result.

*A preliminary version of this work [ADFH18] was presented at the *Genetic and Evolutionary Computation Conference (GECCO) 2018*. In this version, the presentation was improved by rewriting almost the entire text, by giving a clearer comparison with the previous state of the art, by making many proofs more rigorous, by extending the lower bounds to arbitrary fitness functions (subject to a mild restriction on the number of global optima), and by extending our results to the so-called $(N + N)$ EA using a fair parent selection.

1 Introduction

Evolutionary algorithms are general-purpose optimization heuristics and have been successfully applied to a broad range of computational problems. While the majority of the research in evolutionary computation is applied and experimental, the last decades have seen a growing number of theoretical analyses of evolutionary algorithms. Due to the difficult nature of the stochastic processes describing the runs of evolutionary algorithms, the vast majority of these works regards very simple algorithms like the $(1 + 1)$ EA, which has both a parent population and an offspring population of size one. Such works, while innocent looking in their problem statement, can be surprisingly challenging from the mathematical point of view, see, e.g., the long series of works on how the $(1 + 1)$ EA optimizes pseudo-Boolean linear functions which started with the seminal paper [DJW02]. Also, while these example problems are far from the real applications, many of the theoretical works have contributed to the understanding of the working principles of evolutionary algorithms (see, e.g., [Dro05, DHK12, DFK⁺18]), have given advice on how to set parameters and take other design choices (see, e.g., [BDN10, Wit13, RS14]), and even have proposed new algorithms (see, e.g., [DLMN17, DDE15]).

Still, it remains dissatisfying that there are only relatively few works on true population-based algorithms as this bears the risk that we do not really understand the role of populations in evolutionary computation. What is clearly true, and the reason for the lack of such works, is that the stochastic processes become much more complicated when non-trivial populations come into play.

To make some progress towards a better understanding of population-based algorithms, we regard the most simple population-based problem, namely how the elitist $(\mu + \lambda)$ EA optimizes the ONEMAX benchmark problem (see Section 2.2 for the details of this problem). With the corresponding problem for the $(1 + \lambda)$ EA mostly solved in 2005 [JJW05] (see [DK15, Section 8] for a more complete picture) and the problem for the $(\mu + 1)$ EA solved in 2006 [Wit06], it is fair to call this a long-standing open problem. In the conclusion of his paper [Wit06], Witt writes “the most interesting direction seems to be an extension to $(\mu + \lambda)$ strategies by a combination with the existing theory on the $(1 + \lambda)$ EA.”

1.1 Our Results

We give a complete answer to this question and prove that for arbitrary values of μ and λ (which can be functions of the problem size n , however,

for the lower bound we assume that μ is at most polynomial in n , the expected number of iterations the $(\mu + \lambda)$ EA takes to find the optimum of the ONEMAX function is

$$E[T] = \Theta\left(\frac{n \log n}{\lambda} + \frac{n}{\lambda/\mu} + \frac{n \log^+ \log^+(\lambda/\mu)}{\log^+(\lambda/\mu)}\right),$$

where $\log^+ x := \max\{1, \log x\}$ for all $x > 0$. This result subsumes the previous results for the $(1 + \lambda)$ EA and $(\mu + 1)$ EA obtained in [JJW05, DK15, Wit06].

This runtime guarantee shows, e.g., that using a true parent population of size at most $\max\{\log n, \lambda\}$ does not reduce the asymptotic runtime compared to $\mu = 1$. Such information can be useful since it is known that larger parent population sizes can increase the robustness to noise, see, e.g., [GK16].

With our methods, we can also analyze a related algorithm. He and Yao [HY04] and Chen, He, Sun, Chen, and Yao [CHS⁺09] analyzed a version of the $(\mu + \lambda)$ EA in which $\mu = \lambda$ and each parent produces exactly one offspring. We shall call this algorithm the $(\lambda \overset{1:1}{+} \lambda)$ EA for brevity. We prove a tight runtime bound of $\Theta(\frac{n \log n}{\lambda} + n)$ iterations, which also shows that the fairness in the parent selection does not change the asymptotic runtime in this problem.

To prove our bounds, we in particular build on Witt's [Wit03, Wit06, Wit08] family tree argument. The main idea of this argument is to consider a tree graph the vertices of which are the individuals created during the evolution process and each path from the root to a vertex corresponds to the series of mutations which led to the creation of this vertex. Selection does not play any role in this structure, so when working with family trees we usually assume that all individuals with a corresponding vertex in the tree can potentially be present in the current population. Different from Witt's approach (and different from all other works using his method that we are aware of), we work in a complete tree that contains all possible family trees and in this structure argue about which individuals really exist and whether they are an optimal solution. It appears to us that this approach is technically easier than the previous approaches, which first argue that with high probability the true family tree has a certain structure (e.g., a small height) and then, conditioning on this, argue that within such a restricted structure an optimal solution is hard to reach. We also believe that our approach facilitates the uniform analysis of trees having different structures (as, e.g., in our work, in which different relative sizes of μ and λ can lead to very different characteristics of the tree).

Using this argument that does not regard the selection process we have obtained the lower bound that holds not only for the ONEMAX function, but

for any function with a unique optimum in the same way as it was done for the $(\mu + 1)$ EA in [Wit06]. Our arguments let us extend the lower bounds to the functions with multiple optima (however, the number of the optima in these functions must be restricted). In the case of $(\mu + 1)$ EA this extension holds for a broader class of functions than the similar extension in [Wit06] which works only for the functions with a unique optimum.

1.2 Previous Works

The field of mathematical runtime analysis of evolutionary algorithms aims at increasing our understanding via proven results on the performance of evolutionary algorithms. Due to the difficulty of mathematical understanding of complicated population dynamics, the large majority of works in this field considers algorithms with trivial populations. These algorithms may seem trivial, however already allow deep results like the proof of the $O(n \log n)$ expected runtime of the $(1 + 1)$ EA on all linear pseudo-Boolean functions [DJW02, DG13]. They give surprising insights like the fact that monotonic functions can be difficult for simple EAs [DJS⁺13, CDF14, Len18], and have spurred the development of many useful analysis methods [HY01, DJW12].

Despite the mathematical challenges, some results exist on algorithms using non-trivial populations. While such results are quite rare, due the growth of the field in the last 20 years they are still too numerous to be described here exhaustively. Therefore, we describe in the following those results which regard our research problem or special cases of it as well as a few related results.

The two obvious special cases of our problem are runtime analysis of the $(1 + \lambda)$ EA and the $(\mu + 1)$ EA on ONEMAX. In [DK15], the runtime of the $(1 + \lambda)$ EA on the class of linear functions is analyzed, which contains the ONEMAX function. A tight bound of $\Theta\left(\frac{n \log n}{\lambda} + \frac{n \log^+ \log^+ \lambda}{\log^+ \lambda}\right)$ is proven for the expected runtime (number of iterations until the optimum is found) of the $(1 + \lambda)$ EA maximizing the ONEMAX function. This extends the earlier result [JJW05], which shows this bound for $\lambda = O\left(\frac{\log(n) \log \log(n)}{\log \log \log(n)}\right)$, note that in this case the bound simplifies to $\Theta\left(\frac{n \log(n)}{\lambda}\right)$, and which shows further that for asymptotically larger values of λ , the expected runtime is $\omega\left(\frac{n \log(n)}{\lambda}\right)$.

Witt [Wit06] studied the $(\mu + 1)$ EA on the three pseudo-Boolean functions LEADINGONES, ONEMAX and SPC. For the ONEMAX problem, under the mild assumption that μ is polynomially bounded in n , he proved that the expected runtime of the $(\mu + 1)$ EA is $\Theta(\mu n + n \log n)$.

For algorithms with non-trivial parent and offspring population sizes, the following is known. The only work regarding the classic $(\mu + \lambda)$ EA for general μ and λ is [QYZ16]. Using the recent switch analysis technique [YQZ15] and assuming that μ and λ are polynomially bounded in n , it was shown that the $(\mu + \lambda)$ EA needs an expected number of

$$\Omega\left(\frac{n \log n}{\lambda} + \frac{\mu}{\lambda} + \frac{n \log \log n}{\log n}\right)$$

iterations to find the optimum of any function $f : \{0, 1\}^n \rightarrow \mathbb{R}$ with unique optimum. This bound is of smaller asymptotic order (and thus weaker) than ours when $\mu = \omega(\log n)$ and $\frac{\lambda}{\mu} < e^e$ or when $\log \frac{\lambda}{\mu} = \omega(\log n)$, see the discussion at the end of Section 4.

For the $(\lambda \overset{1:1}{+} \lambda)$ EA the first result [HY04, Theorem 4] considers the runtime on the ONEMAX in a special case when $\lambda = n$. It shows an upper bound of $O(n)$ iterations, which is tight as shown by our lower bound.

For the $(\lambda \overset{1:1}{+} \lambda)$ EA with general λ , Chen et al. [CHS⁺09, Proposition 4] show an optimization time of $O(\frac{n \log n}{\lambda} + n \log \lambda)$ iterations. They conjecture a runtime of $O(\frac{n \log n}{\lambda} + n \log \log n)$ [CHS⁺09, Conjecture 3], which is asymptotically at least as good and which is stronger for $\lambda = \omega(\log n)$. Our result improves over this bound and the conjecture for $\lambda = \omega(\frac{\log n}{\log \log n})$ as discussed in Section 6.

We also find notable the result of Dang and Lehre [DL16] (see [ADY19] for recent small improvements) where they proved the upper bound for the (μ, λ) EA on the ONEMAX of $O(n \lambda \log \lambda)$ fitness evaluations when $\lambda > \mu e$ and $\lambda = \Omega(\log(n))$. This runtime can be seen as the upper bound for the $(\mu + \lambda)$ EA, since the population of the $(\mu + \lambda)$ EA is always better than the population of (μ, λ) EA after the same number of iterations in the dominating sense. In Section 3.4 we prove that in the parameters setting regarded in [DL16] our upper bound is asymptotically smaller.

1.3 Organization of the Work

The remainder of the paper is organized as follows. Section 2 gives a formal description of the $(\mu + \lambda)$ EA and introduces the notation that we use in the paper. In Section 3 we prove an upper bound of $O(\frac{n \log n}{\lambda} + \frac{n \mu}{\lambda} + n)$ for the general case and a tighter bound of $O(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}} + \frac{n \log n}{\lambda})$ for the case of $\frac{\lambda}{\mu} > e^e$, when the algorithm is able to gain more than one fitness level during the major part of the optimization process. Section 4 introduces the notion of *complete trees* and proves a lower bound matching our upper bounds. In

Section 5 we extend our lower bounds to a much broader class of functions than just ONEMAX. In Section 6 we provide an analysis of the $(\lambda \overset{1:1}{+} \lambda)$ EA, for which our results cannot be applied directly. The paper ends with a short conclusion and ideas for future work in Section 7.

2 Preliminaries

2.1 Notation

In this subsection we shortly overview the notation used in this paper in order to avoid misunderstanding raised by the plenty of other notations used in the mathematical world nowadays. By \mathbb{N} we denote the set of positive integer numbers and by \mathbb{N}_0 we denote the set of non-negative integer numbers. By $[a..b]$ with $a, b \in \mathbb{N}$ we denote an integer interval which includes its borders. By $[a, b]$ and (a, b) with $a, b \in \mathbb{R} \cup \{-\infty, +\infty\}$ we denote a real-valued interval including and excluding its borders respectively. We denote the binomial distribution with parameters n and p by $\text{Bin}(n, p)$. If some random variable X follows some distribution D , we write $X \sim D$. We use a multiplicative notation of the binomial coefficient, that is

$$\binom{n}{k} = \frac{n(n-1) \dots (n-(k-1))}{k!}.$$

This implies that the binomial coefficients are also defined for $n < k$ and in this case $\binom{n}{k} = 0$.

2.2 Problem Statement

In this section, we provide the definitions necessary to formalize the problem we analyze in this work. Our study focuses on evolutionary algorithms that aim at optimizing pseudo-Boolean functions, that is, functions of the form $f : \{0, 1\}^n \rightarrow \mathbb{R}$.

The $(\mu + \lambda)$ EA formulated as Algorithm 1 is a simple mutation-based elitist evolutionary algorithm. In each iteration of the algorithm, we independently generate λ offspring each by selecting an individual from the parent population uniformly at random and mutating it. We use standard-bit mutation with the standard mutation rate $p = \frac{1}{n}$, that is, we flip each bit independently with probability $\frac{1}{n}$. We note without proof that our results hold as well for any other mutation rate $p = c/n$, where c is a constant.

As objective function f , also called *fitness function*, we consider the classic ONEMAX function, which was the starting point for many theoretical

Algorithm 1: The $(\mu + \lambda)$ EA, maximizing a given function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, with population size μ , offspring population size λ and mutation rate p . We shall exclusively regard the mutation rate $p = \frac{1}{n}$. We did not specify how to break ties in the selection phase since our results are valid for any tie-breaking rule. Usually, one would prefer offspring over parents and break the remaining ties randomly.

1 Initialization:

- 2 Create a population of μ individuals by choosing $x^{(i)} \in \{0, 1\}^n$,
 $1 \leq i \leq \mu$ uniformly at random. Let the multiset
 $X^{(0)} := \{x^{(1)}, \dots, x^{(\mu)}\}$ be the population at time 0. Let $t := 0$.

3 Optimization:

- 4 **while** *an optimum has not been reached* **do**
- 5 $X' := X^{(t)}$;
- 6 **Mutation phase:**
- 7 **for** $i = 1, \dots, \lambda$ **do**
- 8 Choose $x \in X^{(t)}$ uniformly at random;
- 9 Create x' by flipping each bit of x with probability p ;
- 10 $X' := X' \cup \{x'\}$;
- 11 **Selection phase:**
- 12 Create the multiset $X^{(t+1)}$, the population at time $t + 1$, by
deleting the λ individuals with lowest f -value in X' ;
- 13 $t := t + 1$;
-

investigations in this field. This function $\text{ONEMAX} : \{0, 1\}^n \rightarrow \mathbb{R}$ is defined by $\text{ONEMAX}(x) = \sum_{i=1}^n x_i$ for all $x \in \{0, 1\}^n$. In other words, ONEMAX returns the number of one-bits in its argument. Without proof we note that due to the unbiasedness of the operators used by all considered algorithms all our results also hold for the so-called *generalized ONEMAX* function, denoted by ONEMAX_z . This function has some hidden bit-string z and returns the number of coinciding bits in its argument and z . In other words,

$$\text{ONEMAX}_z(x) = \sum_{i=1}^n (1 - |z_i - x_i|) = n - H(x, z),$$

where $H(x, z)$ stands for the Hamming distance.

2.3 Useful Tools

A central argument in our analysis is the following Markov chain argument similar to the classic fitness levels technique of Wegener [Weg01].

Theorem 1. *Let the space S of all possible populations of some population-based algorithm be divided into m disjoint sets A_1, \dots, A_m that are called levels. We write $A_{\geq i} = \bigcup_{j=i}^m A_j$ for all $i \in [1..m]$.*

Let P_t be the population of the algorithm after iteration t . Assume that for all $t \geq 0$ and $i \in [2..m]$, we have that $P_t \in A_i$ implies $\Pr[P_{t+1} \in A_{\geq i}] = 1$. Let T be the minimum number t such that $P_t \in A_m$.

1. *Assume that there are $T_1, \dots, T_{m-1} \geq 0$ such that for all $t \geq 0$ and $i \in [1..m-1]$ we have that if $P_t \in A_i$, then $E[\min\{s \mid s \in \mathbb{N}, P_{t+s} \in A_{\geq i+1}\}] \leq T_i$ (for all possible P_0, \dots, P_{t-1}). Then*

$$E[T] \leq \sum_{i=1}^{m-1} T_i.$$

2. *Assume that there are p_1, p_2, \dots, p_{m-1} such that for all $t \geq 0$ and $i \in [1..m-1]$ we have that if $P_t \in A_i$, then $\Pr[P_{t+1} \in A_{\geq i+1}] \geq p_i$ (for all possible P_0, \dots, P_{t-1}). Then*

$$E[T] \leq \sum_{i=1}^{m-1} \frac{1}{p_i}.$$

The proof is standard, but for the reason of completeness we quickly state it.

Proof. We start by proving the first claim. Consider a run of the algorithm. Let $t_i = \min\{t \mid P_t \in A_{\geq i}\}$. Let $i \in [1..m-1]$. We analyze the random variable $t_{i+1} - t_i$. If there is no t with $P_t \in A_i$, then $t_i = t_{i+1}$ simply by the definition of the t_i . Otherwise, by our assumptions, we have $E[t_{i+1} - t_i] \leq T_i$. Note that this applies trivially also to the first case where we just saw that $t_{i+1} - t_i = 0$. Hence, from $T = t_m = \sum_{i=1}^{m-1} (t_{i+1} - t_i)$ we conclude

$$E[T] = \sum_{i=1}^{m-1} E[t_{i+1} - t_i] \leq \sum_{i=1}^{m-1} T_i.$$

To prove the second claim, we note that by our assumptions $t_{i+1} - t_i$ is stochastically dominated by a geometric distribution with success rate p_i . Hence $E[t_{i+1} - t_i] \leq \frac{1}{p_i}$, and the claim follows as above. \square

To ease the presentation, we use the following language. We say that *the algorithm is on level i* if the current population is in the level A_i . We also say that *the algorithm gains a level* or *the algorithm leaves the current level* if the new population is at the higher level than the previous one.

In our proofs we shall use the following result for random variables with binomial distribution from [GM14]. An elementary proof for it was given in [Doe18].

Lemma 1. *Let $X \sim \text{Bin}(n, p)$ such that $p > 1/n$. Then $\Pr(X \geq E[X]) > 1/4$.*

We also use frequently the following inequality in our proofs, so we formulate it as a separate lemma.

Lemma 2. *For any $x \in (0, 1]$ and any $n > 0$ we have*

$$1 - (1 - x)^n \geq \frac{1}{1 + \frac{1}{xn}}.$$

As was pointed out by one of the reviewers, this lemma is a special case of Lemma 31 in [DL16], which states that for all $n \in \mathbb{N}$ and $x \geq 0$ we have $1 - (1 - x)^n \geq 1 - e^{-xn} \geq \frac{xn}{1+xn}$, except we do not have the constraint that n is an integer. Although the proof of [DL16, Lemma 31] is true for the case $x \in (0, 1]$, we find it wrong for, e.g., $x = 3$ and $n = 2$, when the leftmost part of the inequality is negative, and others are positive. For this reason we show a simple proof here.

Proof. By [RS14, Lemma 8] we have $(1 - x)^n \leq \frac{1}{1+xn}$. Therefore, following the arguments that were used in [RS14, Theorem 9] we conclude

$$\begin{aligned} 1 - (1 - x)^n &\geq 1 - \frac{1}{1 + xn} \\ &= \frac{xn}{1 + xn} = \frac{1}{1 + \frac{1}{xn}}. \end{aligned}$$

□

3 Upper Bounds

In this section, we prove separately two upper bounds for the runtime of the $(\mu + \lambda)$ EA on the ONEMAX problem, the first one being valid for all values of μ and λ and the second one giving an improvement for the case that λ is large compared to μ , more precisely, that $\lambda/\mu \geq e^e$.

Where not specified differently, we denote the current best fitness in the population by i and the number of best individuals in the population by j .

3.1 Increase of the Number of the Best Individuals

In this subsection we analyze how the number of individuals on the current-best fitness level increases over time and derive from this two estimates for the time taken for a fitness improvement. We note that often it is much easier to generate an additional individual with current-best fitness by copying an existing one than to generate an individual having strictly better fitness by flipping the right bits. Consequently, in a typical run of the $(\mu + \lambda)$ EA, first the number of best individuals will increase to a certain number and only then it becomes likely that a strict improvement happens.

Since the increase of the number of individuals on the current-best fitness level via producing copies of such best individuals is independent of the fitness function, we formulate our results for the optimization of an arbitrary pseudo-Boolean function and hope that they might find applications in other runtime analyses as well. So let $f : \{0, 1\}^n \rightarrow \mathbb{R}$ be an arbitrary fitness function which we optimize using the $(\mu + \lambda)$ EA.

Assume that the $(\mu + \lambda)$ EA starts in an arbitrary state where the best individuals have fitness i and there are j_1 such individuals in the population. At this point due to the elitism the algorithm cannot decrease the best fitness i and it also cannot decrease the number of the best individuals j_1 until it increases the best fitness. Following [Sud09, Lemma 2] we call an individual *fit* if it has a fitness i or better. For $j_2 \in \mathbb{N}$, we define $\tau_{j_1, j_2}(i)$ to be the first time (number of iterations) at which the population of the $(\mu + \lambda)$ EA contains at least j_2 fit individuals. We note that this random variable $\tau_{j_1, j_2}(i)$ may depend on the particular initial state of the $(\mu + \lambda)$ EA, but since our results are independent of this initial state (apart from i and j_1) we suppress in our notation the initial state.

The time $\tau_{1, \mu}(i)$, that is, the specific case that $j_1 = 1$ and $j_2 = \mu$, is also called the *takeover time* of a new best individual. For this takeover time, Sudholt [Sud09, Lemma 2] proved the upper bound

$$E[\tau_{1, \mu}(i)] = \lceil \log_5 \mu \rceil \left(\frac{32}{1 - \frac{1}{e}} \cdot \frac{\mu}{\lambda} + 1 \right) = O\left(\frac{\mu \log \mu}{\lambda} + \log \mu \right) \quad (1)$$

for any $i \in [0..n - 1]$.

In this section we improve this result by (i) treating the general case of arbitrary $j_1, j_2 \in [1.. \mu]$ and (ii) by showing an asymptotically smaller bound for the case $\lambda = \omega(\mu)$. In our main analysis of the $(\mu + \lambda)$ EA, we need takeover times for general values of j_2 to profit from the event when we get a fitness gain before the population contains only best individuals, which is likely to happen on the lower fitness levels as we show further. The extension

to general values of j_1 is not needed, but since it does not take extra effort, we do it on the way.

We first prove the following result for arbitrary values of μ and λ . We need this result since it allows arbitrary target numbers j_2 .

Lemma 3. *Let $i \in [0..n - 1]$ and $j_1, j_2 \in [1..\mu]$ with $j_1 < j_2$. Then*

$$E[\tau_{j_1, j_2}(i)] \leq \frac{2e\mu}{\lambda} \left(\ln \frac{j_2}{j_1} + 1 \right) + (j_2 - j_1).$$

Proof. To prove this lemma we use Theorem 1. For this purpose we define levels A_{j_1}, \dots, A_{j_2} . For any $j \in [j_1..j_2 - 1]$ the populations in level A_j have exactly j fit individuals. The level A_{j_2} consists of all populations with at least j_2 fit individuals. Note that the $(\mu + \lambda)$ EA cannot go from level A_j to any other level with smaller index, since it cannot decrease the number of the fit individuals due to the elitist selection.

If there are j fit individuals in the population, then the probability $p_1(j)$ to create as one offspring a copy of a fit individual is the probability to select one of j fit individuals as a parent multiplied by the probability not to flip any bit of it during the mutation. Hence,

$$p_1(j) \geq \frac{j}{\mu} \left(1 - \frac{1}{n} \right)^n \geq \frac{j}{2e\mu}, \quad (2)$$

where we used the inequality $(1 - \frac{1}{n})^n \geq \frac{1}{2e}$ that holds for all $n \geq 2$.

The probability $p_2(j)$ to leave level A_j in one iteration is at least the probability to create a copy of a fit individual as one of the λ offspring. Hence, by Lemma 2 we have

$$p_2(j) \geq 1 - (1 - p_1(j))^\lambda \geq \frac{1}{1 + \frac{1}{p_1(j)\lambda}} \geq \frac{1}{1 + \frac{2e\mu}{j\lambda}}. \quad (3)$$

By Theorem 1 we have

$$E[\tau_{j_1, j_2}(i)] \leq \sum_{j=j_1}^{j_2-1} \frac{1}{p_2(j)} \leq \sum_{j=j_1}^{j_2-1} \left(1 + \frac{2e\mu}{j\lambda} \right) \leq \frac{2e\mu}{\lambda} \left(\ln \frac{j_2}{j_1} + 1 \right) + (j_2 - j_1).$$

□

We note that in case when $j_1 = 1$ and $j_2 = \mu$ our upper bound is $O(\frac{\mu \log \mu}{\lambda} + \mu)$. This is weaker than the upper bound (1) given in [Sud09, Lemma 2] if $\lambda = \omega(\log \mu)$. Without proof we note that in all other cases the two bounds are asymptotically equal.

The reason that our bound is weaker in some cases is that we do not consider the event that the algorithm generates more than one fit offspring in one iteration, while Sudholt in [Sud09, Lemma 2] proved that the number of the fit offspring is multiplied by some constant factor in every $32\mu/\lambda$ iterations. The same idea may be used to prove the bound

$$E[\tau_{j_1, j_2}(i)] \leq \left\lceil \log_5 \frac{j_2}{j_1} \right\rceil \left(\frac{32}{1 - \frac{1}{e}} \cdot \frac{\mu}{\lambda} + 1 \right) = O\left(\frac{\mu \log \frac{j_2}{j_1}}{\lambda} + \log \frac{j_2}{j_1} \right). \quad (4)$$

We still prefer to use to Lemma 3 in our proofs, since it gives us a bound that is easier to operate with due to the simpler leading constants of each term, while the greater terms do not affect our main results.

We now give a second bound for the case that $\frac{\lambda}{\mu} \geq e^e$. It is asymptotically stronger than (1) when $\lambda = \omega(\mu)$ and $\mu = \omega(1)$.

Lemma 4. *Let $\frac{\lambda}{\mu} \geq e^e$. Let $i \in [1..n - 1]$ and $j_1, j_2 \in [1..\mu]$ with $j_1 < j_2$. Then*

$$E[\tau_{j_1, j_2}(i)] \leq 4 \frac{\ln \frac{j_2}{j_1}}{\ln \frac{\lambda}{2e\mu}} + 4.$$

Proof. Let the current population have j fit individuals. Then by (2) the probability that a fixed offspring is a copy of a fit individual is $p_1(j) \geq \frac{j}{2e\mu}$. Therefore, the number N of fit individuals among the λ offspring dominates stochastically a random variable B with binomial distribution $\text{Bin}\left(\lambda, \frac{j}{2e\mu}\right)$. We have $E[B] = \frac{\lambda j}{2e\mu}$. By Lemma 1, $\Pr[B \geq E[B]] \geq \frac{1}{4}$ and thus $\Pr[N \geq \frac{j}{2e\mu}] \geq \frac{1}{4}$. Consequently, in each iteration with probability at least $\frac{1}{4}$ the number of the fit individuals in the population is multiplied by a factor of at least $(1 + \frac{\lambda}{2e\mu})$ (but obviously it cannot become greater than μ).

For a formal proof we define the levels A_1, \dots, A_m , where

$$m := \left\lceil \frac{\ln \frac{j_2}{j_1}}{\ln \left(1 + \frac{\lambda}{2e\mu}\right)} \right\rceil + 1.$$

Level A_m consists of the populations with at least j_2 fit individuals. For $k \in [1..m - 1]$ the populations of level A_k have exactly j fit individuals, where

$$j \in \left[j_1 \left(1 + \frac{\lambda}{2e\mu}\right)^{k-1}, j_1 \left(1 + \frac{\lambda}{2e\mu}\right)^k - 1 \right],$$

and $j < j_2$. To leave any level it is enough to multiply the number of the best individuals by $1 + \frac{\lambda}{2e\mu}$, and the probability of this event is at least $\frac{1}{4}$. By

Theorem 1 we have

$$\begin{aligned} E[\tau_{j_1, j_2}(i)] &\leq \sum_{k=1}^{m-1} 4 = 4 \left\lceil \frac{\ln \frac{j_2}{j_1}}{\ln \left(1 + \frac{\lambda}{2e\mu}\right)} \right\rceil \\ &\leq 4 \frac{\ln \frac{j_2}{j_1}}{\ln \frac{\lambda}{2e\mu}} + 4. \end{aligned}$$

□

We note that the proof of Lemma 4 holds for the weaker assumption $\frac{\lambda}{\mu} > 2e$ as well. However in order not to confuse the reader in Section 3.3 where we consider the case $\frac{\lambda}{\mu} > e^e$ and where this lemma is used, we formulate Lemma 4 with unnecessarily stronger condition.

When $j_2 = \mu$ and $j_1 = 1$ the bound yielded by Lemma 4 is at least as tight as that of (1). For the general values of j_1 and j_2 our bound is at least as tight as the bound (4). When $\lambda/\mu \geq e^e$ the bound (4) simplifies to $O(\log \frac{j_2}{j_1})$. If $\lambda = \omega(\mu)$ and $\frac{j_2}{j_1} = \omega(1)$ then we have

$$4 \frac{\ln \frac{j_2}{j_1}}{\ln \frac{\lambda}{2e\mu}} + 4 = o\left(\log \frac{j_2}{j_1}\right).$$

Therefore, in this case the bound given in Lemma 4 is asymptotically smaller than (4). In all other cases the two bounds are asymptotically equal.

The reason that we have obtained a tighter bound is that we have proven that the number of the fit individuals is multiplied by a more than constant factor with constant probability, while the proof of [Sud09, Lemma 2] considers only the multiplication by a constant factor.

We now use Lemmas 3 and 4 to prove estimates for the time it takes to obtain a strictly better individual once the population contains at least one individual of fitness i . We define \tilde{T}_i as the number of iterations before the algorithm finds an individual with fitness greater than i , if it already has an individual with fitness i in the population. As before, this random variable depends on the precise initial state, but since our results do not rely on the initial state, we suppress it in this notation.

To prove upper bounds on \tilde{T}_i , we estimate the time it takes until some number $\mu_0(i) \in [1.. \mu]$ of individuals with fitness at least i are in the population and then estimate the time to find an improving solution from this situation. We phrase our results here in terms of $\mu_0(i)$ and optimize the value of $\mu_0(i)$ in the later subsections.

Corollary 1. For any $i \in [0..n-1]$ and $\mu_0(i) \in [1..\mu]$, we have

$$E[\tilde{T}_i] \leq \mu_0(i) + \frac{2e\mu}{\lambda}(\ln(\mu_0(i)) + 1) + \frac{e\mu n}{\lambda(n-i)\mu_0(i)}.$$

Proof. Even if the algorithm has only one best individual in the population, in $\tau_{1,\mu_0(i)}(i)$ iterations it will have at least $\mu_0(i)$ individuals with fitness at least i . Assume that at this time we have no individuals with fitness better than i (since otherwise we are done). Let $\tau^+(i)$ be the runtime until the algorithm creates an individual with fitness at least $i+1$ if it already has at least $\mu_0(i)$ individuals with fitness i in the population.

In this setting the probability $p'(i)$ that a particular offspring has fitness better than i is at least the probability to choose one of the $\mu_0(i)$ best individuals and to flip only one of $n-i$ zero-bits in it. We estimate

$$p'(i) \geq \frac{\mu_0(i)(n-i)}{\mu n} \left(1 - \frac{1}{n}\right)^{n-1} \geq \frac{(n-i)\mu_0(i)}{e\mu n}.$$

By Lemma 2 the probability $p''(i)$ to create at least one superior individual among the λ offspring is

$$p''(i) \geq 1 - (1 - p'(i))^\lambda \geq \frac{1}{1 + \frac{1}{\lambda p'(i)}} \geq \frac{1}{1 + \frac{e\mu n}{\lambda(n-i)\mu_0(i)}}. \quad (5)$$

With $p''(i)$ we estimate $E[\tau^+(i)] \leq \frac{1}{p''(i)}$. Therefore, by Lemma 3 we have

$$\begin{aligned} E[\tilde{T}_i] &\leq E[\tau_{1,\mu_0(i)}(i) + \tau^+(i)] = E[\tau_{1,\mu_0(i)}(i)] + E[\tau^+(i)] \\ &\leq E[\tau_{1,\mu_0(i)}(i)] + \frac{1}{p''(i)} \\ &\leq \frac{2e\mu}{\lambda}(\ln \mu_0(i) + 1) + (\mu_0(i) - 1) + 1 + \frac{e\mu n}{\lambda(n-i)\mu_0(i)} \\ &= \mu_0(i) + \frac{2e\mu}{\lambda}(\ln(\mu_0(i)) + 1) + \frac{e\mu n}{\lambda(n-i)\mu_0(i)}. \end{aligned}$$

□

Corollary 2. If $\frac{\lambda}{\mu} > e^e$ then for any $i \in [0..n-1]$ and $\mu_0(i) \in [1..\mu]$, we have

$$E[\tilde{T}_i] \leq 4 \frac{\ln \mu_0(i)}{\ln \frac{\lambda}{2e\mu}} + \frac{e\mu n}{\lambda(n-i)\mu_0(i)} + 5.$$

Proof. Using the same arguments as in the proof of Corollary 1 (in particular, the estimate for $p''(i)$ given in (5)) and by Lemma 4 we estimate

$$E[\tilde{T}_i] \leq E[\tau_{1,\mu_0(i)}(i) + \tau^+(i)] = E[\tau_{1,\mu_0(i)}(i)] + E[\tau^+(i)]$$

$$\begin{aligned}
&\leq E[\tau_{1,\mu_0(i)}(i)] + \frac{1}{p''(i)} \\
&\leq 4 \frac{\ln \mu_0(i)}{\ln \frac{\lambda}{2e\mu}} + 4 + 1 + \frac{e\mu n}{\lambda(n-i)\mu_0(i)} \\
&= 4 \frac{\ln \mu_0(i)}{\ln \frac{\lambda}{2e\mu}} + \frac{e\mu n}{\lambda(n-i)\mu_0(i)} + 5.
\end{aligned}$$

□

We note that Lemma 4 is tight in the sense that we cannot obtain a better upper bound using only the argument of copying the fit individuals.

To formalize this we assume that there is a set $D \subseteq \{0,1\}^n$ of *desired individuals*. We regard the variant EA⁰ of the $(\mu + \lambda)$ EA which only accepts offspring which are desired individuals identical to their parent. Note that the number of desired individuals in a run of this artificial algorithm can never decrease. Assuming that the initial population of the EA⁰ contains exactly j_1 desired individuals, we define $\tau_{j_1, j_2}^*(i)$ as the number of iterations until the population of the EA⁰ contains at least j_2 desired individuals (unlike before, this notation does not depend on the precise initial population as long as it has exactly j_1 desired individuals, but this fact is not important in the following). We show the following result.

Lemma 5. *Let $\frac{\lambda}{\mu} \geq e^e$. Let j_1, j_2 be some integer numbers in $[1..n]$ such that $j_2 > j_1$. Then*

$$E[\tau_{j_1, j_2}^*(i)] = \Omega\left(\frac{\log \frac{j_2}{j_1}}{\log \frac{\lambda}{\mu}} + 1\right).$$

Proof. If $\frac{j_2}{j_1} \leq \frac{\lambda}{\mu}$, then

$$\frac{\log \frac{j_2}{j_1}}{\log \frac{\lambda}{\mu}} + 1 = \Omega(1)$$

and the claim is trivial, since we need at least one iteration to increase the number of the copies in the population.

Consider $\frac{j_2}{j_1} \geq \frac{\lambda}{\mu} \geq e^e$. Let $j(t)$ be the number of the desired individuals after iteration t . We have $j(0) = j_1$. Let $N(t)$ be the number of desired individuals newly created in iteration t . Then $N(t)$ follows a binomial law $\text{Bin}(\lambda, \frac{j(t-1)}{e_n \mu})$, where $e_n := (1 - \frac{1}{n})^{-n} \geq e$. Hence, we have $E[N(t) \mid j(t-1)] = \frac{j(t-1)\lambda}{e_n \mu} \leq \frac{j(t-1)\lambda}{e\mu}$.

For any $t \in \mathbb{N}$ we have $j(t) \leq j(t-1) + N(t)$, where strict inequality occurs only if $j(t-1) + N(t) > \mu$. Therefore, we have

$$E[j(t)] = E[E[j(t) \mid j(t-1)]] \leq E[E[j(t-1) + N(t) \mid j(t-1)]]$$

$$\begin{aligned}
&= E[j(t-1)] + E[E[N(t) \mid j(t-1)]] \leq E[j(t-1)] + E\left[\frac{j(t-1)\lambda}{e\mu}\right] \\
&= E[j(t-1)] + \frac{\lambda}{e\mu} E[j(t-1)] = \left(1 + \frac{\lambda}{e\mu}\right) E[j(t-1)].
\end{aligned}$$

By induction, we obtain

$$E[j(t)] \leq \left(1 + \frac{\lambda}{e\mu}\right)^t j(0) = \left(1 + \frac{\lambda}{e\mu}\right)^t j_1,$$

and by Markov's inequality, we have

$$\Pr[j(t) \geq j_2] \leq \frac{E[j(t)]}{j_2} \leq \left(1 + \frac{\lambda}{e\mu}\right)^t \frac{j_1}{j_2}.$$

For

$$t := \frac{\ln \frac{j_2}{2j_1}}{\ln \left(1 + \frac{\lambda}{e\mu}\right)} = \Omega\left(\frac{\log \frac{j_2}{j_1}}{\log \frac{\lambda}{\mu}}\right)$$

we obtain

$$\Pr[j(t) \geq j_2] \leq \frac{1}{2}.$$

Hence, the probability that the EA⁰ does not obtain j_2 desired individuals in $t = \Omega\left(\frac{\log(j_2/j_1)}{\log(\lambda/\mu)}\right)$ iterations is at least constant. Thus, the expected number of iterations before this happens is at least $\Omega\left(\frac{\log(j_2/j_1)}{\log(\lambda/\mu)}\right)$. \square

3.2 Unconditional Upper Bound

Having the results of Section 3.1 we first prove the following upper bound, which is valid for all values of μ and λ . When λ is not significantly larger than μ , then the $(\mu + \lambda)$ EA typically increases the best fitness by at most a constant in each iteration. For this reason, we can use Theorem 1 and obtain a runtime bound that will turn out to be tight for this case.

Theorem 2. *The expected number of iterations for the $(\mu + \lambda)$ EA to optimize the ONEMAX problem is*

$$O\left(\frac{n \log n}{\lambda} + \frac{n\mu}{\lambda} + n\right).$$

Proof. To use Theorem 1 we define levels A_0, \dots, A_n such that level A_i , $i \in [0..n]$, consists of all populations having maximum fitness equal to i . In Corollary 1 we have already estimated the expected times $E[\tilde{T}_i]$ the

$(\mu + \lambda)$ EA takes to leave these levels. These estimates depended on the number $\mu_0(i)$ of individuals of fitness i we aim at before leaving the level. By choosing suitable values for the $\mu_0(i)$ we prove our bound.

The choice of $\mu_0(i)$ is guided by the following trade-off. If we choose $\mu_0(i) = \mu$, then after having μ best individuals in the population we have the highest probability to find a better individual. However, we pay for it with the time we spend on obtaining μ copies of the best individual. On the other hand, if we choose $\mu_0(i) = 1$ we do not spend any iteration filling the population with copies of the best individual, but we have a low chance to increase the current fitness. How this trade-off is optimally resolved, and hence the optimal value of $\mu_0(i)$, depends on the probability to create a better individual and thus on the current fitness i .

We distinguish three cases depending on current fitness i . The “milestones” which mark the transition between these cases are the fitness values $i = \lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil$ and $i = \lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor$. While the best fitness is below the first milestone, the probability to increase the fitness is so high that we do not need to have more than one best individual in the population. Beyond the second milestone this probability is so low that we better spend the time to fill the population with the copies of the best individual. Between the two milestones we have to find a suitable value of $\mu_0(i)$ to give a balanced trade-off.

To simplify the notation, we define $\Delta_i := 1 + \sqrt{1 + \frac{n\lambda}{e(n-i)\mu}}$. Note that

$$1 + \sqrt{\frac{n}{n-i}} \sqrt{\frac{\lambda}{e\mu}} \leq \Delta_i \leq 1 + \sqrt{\frac{n}{n-i}} \sqrt{1 + \frac{\lambda}{e\mu}}. \quad (6)$$

This value of Δ_i arises from the computation of the derivative of the upper bound on $E[\tilde{T}_i]$ from Corollary 1, which is needed to find the optimal value of $\mu_0(i)$ in the second case, when the current fitness is between the two milestones.

For $i \leq \lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil$ we define $\mu_0(i) := 1$. By Corollary 1 we have

$$E[\tilde{T}_i] \leq \frac{e\mu n}{\lambda(n-i)} + \frac{2e\mu}{\lambda} + 1.$$

Let T_1 be the number of iterations before the $(\mu + \lambda)$ EA finds an individual with fitness greater than $\lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil$ for the first time. Then by Theorem 1 we have

$$E[T_1] \leq \sum_{i=0}^{\lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil} E[\tilde{T}_i] \leq \sum_{i=0}^{\lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil} \left(\frac{e\mu n}{\lambda(n-i)} + \frac{2e\mu}{\lambda} + 1 \right)$$

$$\begin{aligned}
&\leq \frac{e\mu n}{\lambda} \left(\ln(n) - \ln\left(\frac{n}{2 + \lambda/e\mu}\right) + 1 \right) + \frac{2e\mu}{\lambda}n + n \\
&= \frac{e\mu n}{\lambda} \ln\left(2 + \frac{\lambda}{e\mu}\right) + \frac{3e\mu n}{\lambda} + n \\
&= O\left(\frac{\mu n}{\lambda}\right) + O(n),
\end{aligned}$$

where we used the estimate $\frac{e\mu}{\lambda} \ln(2 + \frac{\lambda}{e\mu}) = O(1 + \frac{\mu}{\lambda})$ that holds for any asymptotic behavior of μ/λ .

For $\lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil < i \leq \lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor$ **we define** $\mu_0(i) := \lceil \frac{n}{(n-i)\Delta_i} \rceil$. By Corollary 1 we have

$$E[\tilde{T}_i] \leq \frac{n}{(n-i)\Delta_i} + 1 + \frac{2e\mu}{\lambda} \left(\ln \frac{n}{(n-i)\Delta_i} + 2 \right) + \frac{e\mu\Delta_i}{\lambda}.$$

By (6), we have

$$\begin{aligned}
E[\tilde{T}_i] &\leq \frac{n}{(n-i) \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{\frac{\lambda}{e\mu}}\right)} + 1 \\
&\quad + \frac{2e\mu}{\lambda} \left(\ln \frac{n}{(n-i) \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{\frac{\lambda}{e\mu}}\right)} + 2 \right) \\
&\quad + \frac{e\mu}{\lambda} \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{1 + \frac{\lambda}{e\mu}} \right).
\end{aligned} \tag{7}$$

Let T_2 be the number of iterations until the $(\mu + \lambda)$ EA finds an individual with fitness greater than $\lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor$ for the first time if it already has an individual with fitness greater than $\lceil n - \frac{n}{2+\lambda/(e\mu)} \rceil$ in the population. By

Theorem 1 and by (7), we obtain

$$\begin{aligned}
E[T_2] &\leq \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} E[\tilde{T}_i] \\
&\leq n \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \frac{1}{(n-i) \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{\frac{\lambda}{e\mu}}\right)} \\
&\quad + \frac{e\mu}{\lambda} \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{1 + \frac{\lambda}{e\mu}}\right) \\
&\quad + \frac{2e\mu}{\lambda} \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \ln \left(\frac{n}{(n-i)\Delta_i} \right) + \frac{2e\mu}{\lambda} 2n + n.
\end{aligned} \tag{8}$$

We regard three sums in (8) separately. First, by the estimate $\sum_{i=1}^n 1/\sqrt{i} \leq 1 + \int_1^n (1/\sqrt{x})dx < 2\sqrt{n}$, we obtain

$$\begin{aligned}
&\sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \frac{1}{(n-i) \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{\frac{\lambda}{e\mu}}\right)} \\
&\leq \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \frac{1}{(n-i) \sqrt{\frac{n}{n-i}} \sqrt{\frac{\lambda}{e\mu}}} \\
&= \sqrt{\frac{e\mu}{\lambda n}} \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \frac{1}{\sqrt{n-i}} \\
&\leq \sqrt{\frac{e\mu}{\lambda n}} \cdot 2\sqrt{\frac{n}{2+\lambda/(e\mu)}} \leq 2\sqrt{\frac{e\mu}{\lambda n}} \sqrt{\frac{n}{\lambda/(e\mu)}} = 2e\frac{\mu}{\lambda}.
\end{aligned} \tag{9}$$

To analyze the second sum we also use the estimate $\frac{1+t}{2+t} < 1$ valid for all $t \in [0, +\infty)$. We obtain

$$\begin{aligned}
& \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \left(1 + \sqrt{\frac{n}{n-i}} \sqrt{1 + \frac{\lambda}{e\mu}} \right) \\
& \leq n + \sqrt{n \left(1 + \frac{\lambda}{e\mu} \right)} \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \frac{1}{\sqrt{n-i}} \\
& \leq n + \sqrt{n \left(1 + \frac{\lambda}{e\mu} \right)} \cdot 2 \sqrt{\frac{n}{2 + \lambda/(e\mu)}} \\
& = n + 2n \sqrt{\frac{1 + \lambda/(e\mu)}{2 + \lambda/(e\mu)}} \leq 3n.
\end{aligned} \tag{10}$$

For the last sum we use the logarithmic version of Stirling's formula, that is, $\ln(n!) = n \ln(n) - n + O(\log(n))$ (see, e.g. [Rob55] or [Doe20, Theorem 1.4.10]), and the estimate $\frac{\ln(2+t)+2}{2+t} \leq 2$ for all $t \in [0, +\infty)$. We obtain

$$\begin{aligned}
& \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \ln \left(\frac{n}{(n-i)\Delta_i} \right) \\
& \leq \sum_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil+1}^{\lfloor n-\frac{n}{\mu(2+\lambda/e)} \rfloor} \ln \left(\frac{n}{(n-i)} \right) \\
& \leq \left\lceil \frac{n}{2 + \lambda/(e\mu)} \right\rceil \ln(n) - \ln \left(\prod_{i=\lceil n-\frac{n}{2+\lambda/(e\mu)} \rceil}^n (n-i) \right) \\
& \leq \left\lceil \frac{n}{2 + \lambda/(e\mu)} \right\rceil \ln(n) - \ln \left(\left\lceil \frac{n}{2 + \lambda/(e\mu)} \right\rceil! \right) \\
& = \left\lceil \frac{n}{2 + \lambda/(e\mu)} \right\rceil \left(\ln(n) - \ln \left\lceil \frac{n}{2 + \lambda/(e\mu)} \right\rceil + 1 \right) \\
& \quad + O \left(\log \left\lceil \frac{n}{2 + \lambda/(e\mu)} \right\rceil \right) \\
& \leq \frac{n}{2 + \lambda/(e\mu)} \left(\ln \left(2 + \frac{\lambda}{e\mu} \right) + 2 \right) + o(n) \\
& \leq 2n + o(n).
\end{aligned} \tag{11}$$

Finally, by putting (9), (10) and (11) into (8) we obtain

$$\begin{aligned} E[T_2] &\leq n \cdot 2e \frac{\mu}{\lambda} + \frac{e\mu}{\lambda} \cdot 3n + \frac{2e\mu}{\lambda} (2n + o(n)) + \frac{4e\mu n}{\lambda} + n \\ &= \frac{13e\mu n}{\lambda} + n + o\left(\frac{\mu n}{\lambda}\right) = O\left(\frac{\mu n}{\lambda} + n\right). \end{aligned}$$

For $n - 1 \geq i > \lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor$ **we define** $\mu_0(i) := \mu$. Note that this case can only appear when $\frac{n}{\mu(2+\lambda/e)} \geq 1$ and thus $\mu \leq \frac{n}{(2+\lambda/e)} = O(n/\lambda)$. By Corollary 1 the expected waiting time for a fitness gain is at most

$$E[\tilde{T}_i] \leq \mu + \frac{2e\mu}{\lambda} (\ln(\mu) + 1) + \frac{en}{\lambda(n-i)}.$$

Let T_3 be the number of iterations until the $(\mu + \lambda)$ EA finds the optimum starting from the moment when it has an individual with fitness greater than $\lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor$ in the population. Then by Theorem 1 we have

$$\begin{aligned} E[T_3] &\leq \sum_{i=\lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor + 1}^{n-1} E[\tilde{T}_i] \\ &\leq \sum_{i=\lfloor n - \frac{n}{\mu(2+\lambda/e)} \rfloor + 1}^{n-1} \left(\mu + \frac{2e\mu}{\lambda} (\ln(\mu) + 1) + \frac{en}{\lambda(n-i)} \right) \\ &\leq \mu \frac{n}{\mu(2+\lambda/e)} + \frac{2e\mu}{\lambda} (\ln(\mu) + 1) \frac{n}{\mu(2+\lambda/e)} \\ &\quad + \frac{en}{\lambda} \left(\ln \frac{n}{\mu(2+\lambda/e)} + 1 \right) \\ &= O\left(\frac{n}{\lambda}\right) + O\left(\frac{n \log \mu}{\lambda^2}\right) + O\left(\frac{n \log n}{\lambda}\right) \\ &= O\left(\frac{n \log \mu}{\lambda^2}\right) + O\left(\frac{n \log n}{\lambda}\right) = O\left(\frac{\mu n}{\lambda}\right) + O\left(\frac{n \log n}{\lambda}\right). \end{aligned}$$

Summing the expected runtimes for all cases, we obtain the upper bound for the expected total runtime.

$$\begin{aligned} E[T] &\leq E[T_1] + E[T_2] + E[T_3] \\ &= O\left(\frac{\mu n}{\lambda} + n\right) + O\left(\frac{\mu n}{\lambda} + n\right) + O\left(\frac{\mu n}{\lambda} + \frac{n \log n}{\lambda}\right) \\ &= O\left(\frac{n \log n}{\lambda} + \frac{\mu n}{\lambda} + n\right). \end{aligned}$$

□

3.3 Upper Bound with Large λ

In this section we consider the case when $\frac{\lambda}{\mu} > e^e$. Due to the large number of offspring the algorithm performs significantly better in this case. The first reason of this speed-up is that the algorithm can now gain several fitness levels in one iteration with high probability when the current-best fitness is small. The second reason is the faster increase of the number of best individuals, see Corollary 2.

These two observations allow us to prove the following upper bound on the runtime.

Theorem 3. *If $\frac{\lambda}{\mu} \geq e^e$ then the expected number of iterations for the $(\mu + \lambda)$ EA to optimize the ONEMAX problem is*

$$O\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}} + \frac{n \log n}{\lambda}\right).$$

Note that the bound given in Theorem 3 is asymptotically the same as the bound given in Theorem 2 when $\frac{\lambda}{\mu} = \Theta(1)$. The difference between the two bounds becomes asymptotically significant only when $\frac{\lambda}{\mu} = \omega(1)$. Therefore it does not matter which constant we choose to distinguish the fast regime of the algorithm. The main purpose of the choice of e^e as a border value is to simplify the proofs and to improve their readability. However without proof we note that all arguments used in this section hold also for the smaller values of $\frac{\lambda}{\mu}$ which are greater than $2e$.

To prove Theorem 3 we split the optimization process into four phases. Each phase corresponds to some range of the best fitness values, and the phase transition occurs at fitness values $n - \frac{n}{\ln \frac{\lambda}{\mu}}$, $n - \frac{\mu n}{\lambda}$ and $n - \frac{n}{\lambda}$. In each phase the $(\mu + \lambda)$ EA has a specific behavior, so we analyze each phase separately in the following four lemmas.

During the first phase, while the fitness of the best individual is below $n - \frac{n}{\ln \frac{\lambda}{\mu}}$, regardless of the number of best individuals, with constant probability we generate an offspring increasing the best fitness in the population by at least $\gamma := \lfloor \frac{\ln \frac{\lambda}{\mu}}{2 \ln \ln \frac{\lambda}{\mu}} \rfloor$. So we need not more than an expected number of $O(\frac{n}{\gamma})$ iterations to finish the first phase.

Let R_1 be the runtime of the $(\mu + \lambda)$ EA until it finds an individual with fitness at least $n - \frac{n}{\ln \frac{\lambda}{\mu}}$, in other words, the duration of the first phase. We prove the following upper bound on the expected value of R_1 .

Lemma 6 (Phase 1). *If $\frac{\lambda}{\mu} \geq e^e$, then we have*

$$E[R_1] = O\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right).$$

Proof. To use Theorem 1, we split the space of populations S into levels A_1, \dots, A_m , where

$$m := \left\lceil \frac{\lfloor n - \frac{n}{\ln \frac{\lambda}{\mu}} \rfloor}{\gamma} \right\rceil + 1.$$

If $k < m$, then the populations of level A_k have the fitness of the best individual in $[(k-1)\gamma, k\gamma-1]$ (but less than $n - \frac{n}{\ln \frac{\lambda}{\mu}}$). The level A_m consists of all populations containing an individual of fitness at least $n - \frac{n}{\ln \frac{\lambda}{\mu}}$.

To show that we have a constant probability to leave any level, we consider the probability that a particular offspring has a fitness exceeding the current best fitness i by at least γ . This is at least the probability to choose one of the best individuals and to flip exactly γ zero-bits in it and not to flip the other $n - \gamma$ bits, namely

$$\binom{n-i}{\gamma} \frac{j}{\mu n^\gamma} \left(1 - \frac{1}{n}\right)^{n-\gamma} \geq \frac{j}{e\mu} \left(\frac{n-i}{n\gamma}\right)^\gamma =: p_\gamma(i).$$

The probability to increase the best fitness by at least γ with one of λ offspring is at least $1 - (1 - p_\gamma(i))^\lambda$. Thus, by Lemma 2, the expected number of iterations for this to happen is not larger than

$$\frac{1}{1 - (1 - p_\gamma(i))^\lambda} \leq 1 + e \frac{\mu}{\lambda} \left(\frac{n\gamma}{n-i}\right)^\gamma.$$

Since $\frac{\lambda}{\mu} \geq e^e$, we have $\gamma = \lfloor \frac{\ln \frac{\lambda}{\mu}}{2 \ln \ln \frac{\lambda}{\mu}} \rfloor \geq \lfloor \frac{e}{2} \rfloor = 1$. Using this and the estimate $\frac{n}{n-i} \leq \ln \frac{\lambda}{\mu}$ valid during this phase, we compute

$$\begin{aligned} \left(\frac{n\gamma}{n-i}\right)^\gamma &\leq \exp\left(\gamma \ln\left(\gamma \ln \frac{\lambda}{\mu}\right)\right) \leq \exp\left(\frac{\ln \frac{\lambda}{\mu}}{2 \ln \ln \frac{\lambda}{\mu}} \ln\left(\frac{\ln^2 \frac{\lambda}{\mu}}{2 \ln \ln \frac{\lambda}{\mu}}\right)\right) \\ &\leq \exp\left(\frac{\ln \frac{\lambda}{\mu}}{2 \ln \ln \frac{\lambda}{\mu}} 2 \ln \ln \frac{\lambda}{\mu}\right) = \exp\left(\ln \frac{\lambda}{\mu}\right) = \frac{\lambda}{\mu}. \end{aligned}$$

Therefore, the expected time to increase the fitness by γ (and thus to leave level A_k for any $k < m$) is at most $1 + e$. Summing over the levels

A_1, \dots, A_{m-1} , by Theorem 1 we have

$$E[R_1] \leq \sum_{k=1}^{m-1} (1+e) < (1+e)m < (1+e)\frac{n}{\gamma} = O\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right).$$

□

Having found an individual with fitness at least $n - \frac{n}{\ln \frac{\lambda}{\mu}}$, we enter the second phase. Due to the elitist selection, the minimum fitness in the population does not decrease, so there is no risk of a fall-back into the first phase.

In the second phase, due to the smaller distance from the optimum, fitness gains by more than a constant are too rare to be exploited profitably. However, even when we only have one best individual in the population, the probability to create at least one better individual in one iteration will still be constant. Consequently, we do not need the arguments of Section 3.1 analyzing how the the number of best individuals grows. This phase ends when the best fitness in the population is $n - \frac{\mu n}{\lambda}$ or more.

Let R_2 be the runtime of the $(\mu + \lambda)$ EA until it finds an individual with fitness at least $n - \frac{\mu n}{\lambda}$ starting from the moment when it has an individual with fitness at least $n - \frac{n}{\ln \frac{\lambda}{\mu}}$ in the population. In other words, R_2 is the duration of the second phase.

Lemma 7 (Phase 2). *If $\frac{\lambda}{\mu} \geq e^e$, then we have*

$$E[R_2] = O\left(\frac{n}{\log \frac{\lambda}{\mu}}\right).$$

Proof. For

$$i \in \left[\left\lceil n - \frac{n}{\ln \frac{\lambda}{\mu}} \right\rceil .. \left\lceil n - \frac{\mu n}{\lambda} \right\rceil - 1 \right],$$

the level B_i is defined as the set of all populations in which the best individuals have fitness i . For $i = \lceil n - \frac{\mu n}{\lambda} \rceil$ let the level B_i consist of all populations with best fitness at least i .

By Corollary 2 and defining $\mu_0(i) := 1$ for all i , we have

$$E[\tilde{T}_i] \leq \frac{e\mu n}{\lambda(n-i)} + 5 \leq e + 5,$$

where the last estimate follows from $i \leq n - \frac{\mu n}{\lambda}$. Therefore, by Theorem 1

$$E[R_2] \leq \sum_{i=\lceil n - \frac{\mu n}{\lambda} \rceil}^{\lceil n - \frac{\mu n}{\lambda} \rceil - 1} E[\tilde{T}_i] \leq (5+e)\frac{n}{\ln \frac{\lambda}{\mu}} = O\left(\frac{n}{\log \frac{\lambda}{\mu}}\right).$$

□

After completion of the second phase, generating a strictly better individual is so difficult that it pays off (in the analysis) to wait for more than one best individual in the population. More precisely, depending on the current best fitness i we define a number $\mu_0(i)$ and compute the time to reach $\mu_0(i)$ best individuals and argue that the expected time to generate a strict improvement when at least $\mu_0(i)$ best individuals are in the population is only constant. Since, as discussed in Section 3.1, specifically in Lemma 4, the number of the best individuals in the population roughly increases by a factor $(1 + \frac{\lambda}{2e\mu})$ in each iteration, the algorithm obtains $\mu_0(i)$ individuals reasonably fast.

Let R_3 be the runtime of the $(\mu + \lambda)$ EA until it finds an individual with fitness at least $n - \frac{n}{\lambda}$, the end of the third phase, starting from the moment when it has an individual with fitness at least $n - \frac{\mu n}{\lambda}$ in the population.

Lemma 8 (Phase 3). *If $\frac{\lambda}{\mu} \geq e^e$, then we have*

$$E[R_3] = O\left(\frac{\mu n}{\lambda}\right).$$

Proof. During this phase the best fitness i in the population satisfies

$$n - \frac{\mu n}{\lambda} \leq i < n - \frac{n}{\lambda},$$

which implies

$$\frac{\lambda}{\mu} \leq \frac{n}{n-i} < \lambda. \quad (12)$$

For these values of i we define $\mu_0(i) := \lceil \frac{n\mu}{(n-i)\lambda} \rceil$. Note that $\mu_0(i) \in [1.. \mu]$.

For

$$i \in \left[\lceil n - \frac{\mu n}{\lambda} \rceil .. \lceil n - \frac{n}{\lambda} \rceil - 1 \right],$$

level C_i is defined as a set of all populations in which the best individuals have fitness i . For $i = \lceil n - \frac{n}{\lambda} \rceil$ let the level C_i consist of all populations with best fitness at least i .

By Corollary 2 and by the definition of $\mu_0(i)$ we have

$$\begin{aligned} E[\tilde{T}_i] &\leq 4 \frac{\ln \mu_0(i)}{\ln \frac{\lambda}{2e\mu}} + \frac{e\mu n}{\lambda(n-i)\mu_0(i)} + 5 \\ &\leq \frac{4}{\ln \frac{\lambda}{2e\mu}} \left(\ln \frac{n\mu}{(n-i)\lambda} + 1 \right) + e + 5. \end{aligned}$$

By Theorem 1 we obtain

$$\begin{aligned}
E[R_3] &\leq \sum_{i=\lceil n-n\mu/\lambda \rceil}^{\lceil n-n/\lambda \rceil-1} \tilde{T}_i \\
&\leq \sum_{i=\lceil n-n\mu/\lambda \rceil}^{\lceil n-n/\lambda \rceil-1} \left(\frac{4}{\ln \frac{\lambda}{2e\mu}} \left(\ln \frac{n\mu}{(n-i)\lambda} + 1 \right) + e + 5 \right) \\
&\leq \frac{4}{\ln \frac{\lambda}{2e\mu}} \left(\frac{n\mu}{\lambda} + \sum_{i=\lceil n-n\mu/\lambda \rceil}^{\lceil n-n/\lambda \rceil-1} \ln \frac{n\mu}{(n-i)\lambda} \right) + \frac{n\mu}{\lambda}(e+5).
\end{aligned}$$

We estimate $\sum_{i=\lceil n-n\mu/\lambda \rceil}^{\lceil n-n/\lambda \rceil-1} \ln \frac{n\mu}{(n-i)\lambda}$ using Stirling's formula as in (11). We also notice that this phase occurs only when $\frac{n\mu}{\lambda} > 1$, thus we have $(\ln \frac{n\mu}{\lambda} - \ln \lfloor \frac{n\mu}{\lambda} \rfloor) \leq 1$. Hence, we obtain.

$$\begin{aligned}
\sum_{i=\lceil n-n\mu/\lambda \rceil}^{\lceil n-n/\lambda \rceil-1} \ln \frac{n\mu}{(n-i)\lambda} &\leq \sum_{i=1}^{\lfloor n\mu/\lambda \rfloor} \ln \frac{n\mu}{i\lambda} \\
&= \left\lfloor \frac{n\mu}{\lambda} \right\rfloor \ln \frac{n\mu}{\lambda} - \left\lfloor \frac{n\mu}{\lambda} \right\rfloor \ln \left\lfloor \frac{n\mu}{\lambda} \right\rfloor \\
&\quad + \left\lfloor \frac{n\mu}{\lambda} \right\rfloor + O\left(\log \left\lfloor \frac{n\mu}{\lambda} \right\rfloor\right) \\
&= \left\lfloor \frac{n\mu}{\lambda} \right\rfloor \left(\ln \frac{n\mu}{\lambda} - \ln \left\lfloor \frac{n\mu}{\lambda} \right\rfloor + 1 \right) + o\left(\frac{\mu n}{\lambda}\right) \\
&\leq 2\frac{n\mu}{\lambda} + o\left(\frac{\mu n}{\lambda}\right).
\end{aligned}$$

Therefore,

$$E[R_3] \leq (5+e)\frac{\mu n}{\lambda} + 4\frac{2\frac{n\mu}{\lambda} + o\left(\frac{\mu n}{\lambda}\right) + \frac{n\mu}{\lambda}}{\ln \frac{\lambda}{2e\mu}} = O\left(\frac{\mu n}{\lambda}\right).$$

□

When the algorithm is closer to the optimum than in the third phase, then we cannot expect to have a constant probability for a strict fitness improvement even when the whole population consists of individuals of best fitness. In this forth and last phase, we thus always wait (in the analysis) until the population only contains best individuals and then estimate the expected time for an improvement. We denote by R_4 the runtime until the algorithm finds the optimum if it already has an individual with fitness at least $n - \frac{n}{\lambda}$ in the population.

Lemma 9 (Phase 4). *If $\frac{\lambda}{\mu} \geq e^e$ then*

$$E[R_4] = O\left(\frac{n \log n}{\lambda}\right).$$

Proof. For

$$i \in \left[\left[n - \frac{n}{\lambda} \right] .. n - 1 \right]$$

we define level D_i as a set of all populations in which the best individuals have fitness i . We also define $\mu_0(i) = \mu$ for these values of i .

By Corollary 2 we have

$$E[\tilde{T}_i] \leq 4 \frac{\ln \mu}{\ln \frac{\lambda}{2e\mu}} + \frac{en}{\lambda(n-i)} + 5.$$

Therefore, by Theorem 1, we obtain

$$\begin{aligned} E[R_4] &\leq \sum_{i=\lceil n-\frac{n}{\lambda} \rceil}^{n-1} \left(\frac{4 \ln \mu}{\ln \frac{\lambda}{2e\mu}} + \frac{en}{\lambda(n-i)} + 5 \right) \\ &\leq \frac{4n \ln \mu}{\lambda \ln \frac{\lambda}{2e\mu}} + \frac{en(\ln \frac{n}{\lambda} + 1)}{\lambda} + \frac{5n}{\lambda} \\ &= O\left(\frac{n}{\log \frac{\lambda}{\mu}}\right) + O\left(\frac{n \log n}{\lambda}\right). \end{aligned}$$

□

Finally, we prove Theorem 3.

Proof (Theorem 3). Since we consider an elitist algorithm that cannot reduce the best fitness, by linearity of expectation and Lemmas 6 to 9 we have

$$E[T] \leq E[R_1] + E[R_2] + E[R_3] + E[R_4] = O\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}} + \frac{n \log n}{\lambda}\right).$$

□

3.4 Comparison With Other Upper Bounds

We first note that our upper bound

$$O\left(\frac{n \log n}{\lambda} + \frac{n}{\lambda/\mu} + \frac{n \log^+ \log^+(\lambda/\mu)}{\log^+(\lambda/\mu)}\right)$$

for the runtime of the $(\mu + \lambda)$ EA on ONEMAX subsumes the known bounds

$$O(n \log n + \mu n)$$

for the $(\mu + 1)$ EA [Wit06] and

$$O\left(\frac{n \log n}{\lambda} + \frac{n \log^+ \log^+ \lambda}{\log^+ \lambda}\right)$$

for the $(1 + \lambda)$ EA [DK15].

We are not aware of any previous result for the $(\mu + \lambda)$ EA for general values of μ and λ . Using a standard domination argument, however, the results of Dang and Lehre [DL16] for the (μ, λ) EA can be extended to the $(\mu + \lambda)$ EA. Recall that the (μ, λ) EA differs from the $(\mu + \lambda)$ EA only in the selection mechanism, which disallows the (μ, λ) EA to select any parent individual into the next population, even the ones which are better than all λ offspring. This imposes a constraint on the parameters requiring λ to be at least μ . For the case that $\lambda > (1 + \varepsilon)e\mu$, $\varepsilon > 0$ a constant¹, and $\lambda = \Omega(\log n)$, Dang and Lehre [DL16, Theorem 14] proved that the (μ, λ) EA within an expected number of

$$O(n \log \lambda)$$

iterations finds the optimum of ONEMAX.

Since the $(\mu + \lambda)$ EA uses elitist selection, it can be shown that the fitness values of its population always stochastically dominate those of the population of the (μ, λ) EA. More precisely, for a run of the $(\mu + \lambda)$ EA let us for $i \in [1..\mu]$ and $t \in \mathbb{N}$ denote by f_{it} the fitness of the i -th individual in the parent population after iteration t , where we assume that the individuals are sorted by decreasing fitness. Let us denote by f'_{it} the same for the (μ, λ) EA. Then for all i and t , the random variable f_{it} stochastically dominates f'_{it} . This can be shown via coupling in a similar fashion as in the proof of Theorem 23 in [Doe19]. Thus the upper bound given by Dang and Lehre is also valid for the $(\mu + \lambda)$ EA. For the case $\lambda > (1 + \varepsilon)e\mu$ and $\lambda = \Omega(\log n)$ regarded by Dang and Lehre, our bound becomes

$$O(n),$$

which is of an asymptotically slightly smaller order than that of [DL16].

¹In [DL16] only $\lambda > e\mu$ is required, but from analyzing their proof we suspect that the stronger condition is implicitly used.

4 Lower Bounds

In this section, we show the lower bounds corresponding to the upper bounds we proved in the previous section. They in particular imply the lower bounds for the $(\mu + 1)$ EA given in [Wit06] and the $(1 + \lambda)$ EA given in [DK15]. Hence our proof method is a unified approach to both these algorithms as well. The arguments we use do not consider selection phase at all, thus they hold also for all functions with a unique optimum and for other selection mechanisms, including the (μ, λ) EA.

The main problem when proving lower bounds for population-based algorithms is that many individuals which are created during the run of the EA are removed at some stage by selection operations. This creates a complicated population dynamics, which is very hard to follow via mathematical means.

One way to overcome this difficulty is to try to disregard the effect of selection and instead regard an optimistic version of the evolutionary process in which no individuals are removed. This idea can be traced back to [RRS98]. In the context of evolutionary computation, it has been first used in [Wit03] (see [Wit08] for an extended version) in the analysis of a steady-state genetic algorithm with fitness-proportionate selection. In [JW05], this argument was used in the analysis of a $(\mu + 1)$ evolution strategy (in continuous search spaces). Not surprisingly, the analysis of the $(\mu + 1)$ EA [Wit06] uses the artificial populations argument as well.

This technique then found applications in the analysis of memetic algorithms [Sud09], aging-mechanisms [JZ11], and non-elitist algorithms [Leh10, LY12]. The artificial population argument was also used to overcome the difficulties imposed by another removal mechanism, namely Pareto domination in evolutionary multi-objective optimization [DKV13]. While similar in spirit, this work however uses quite different techniques, e.g., it does not represent the search process via tree structures.

Of course, to make the new process really an optimistic version of the original one, we have to ensure that, despite the larger population present, each individual which is also present in the true population has the same power of creating good solutions as in the original process. To ensure this in our process, we assume that in the artificial process each individual creates $\text{Bin}(\lambda, 1/\mu)$ offspring. This assumption, in fact, leads to a much more drastic growth of the artificial population than the fact that we disregard selection.

When working with such an artificially enlarged population, there is a risk that the larger population finds it easier to create the optimal solution. This would give weaker lower bounds. So the main art in this proof approach is setting up the arguments in a way that the larger population does still, in

an asymptotic sense, not find the optimum earlier than the original process. The reason why this is possible at all is that once selection is disregarded, the process consists only of independent applications of the mutation operator. This allows to use strong-concentration arguments which in the end give the desired result that none of the many members of the artificial population is the optimal solution.

To make this approach formal, we use the following notion of a *complete tree*, which, in simple words, describes all possible (iterated) offspring which could occur in a run of the evolutionary algorithm. This notion is different from those used in the works above, which all work with certain subtrees of the complete tree and use suitable arguments to reason that the restricted tree still covers all individuals that can, with reasonable probability, appear. We feel that our approach of working in the complete tree is technically simpler. For example, compared to [Wit06], we do not first need to argue that with high probability the true tree has only certain depths and then, conditional on this event, argue that it does not contain an optimal solution. Working in the complete tree, we also do not need arguments from branching processes as used in [LY12]. Of course, the key argument that without selection we only do repeated unguided mutation, is used by us in the same flavor as in all previous works.

More precisely, the complete tree with initial individual x_0 is defined recursively as follow. Every vertex is labeled with some individual (a bit-string) which could potentially occur in the evolution process. The labels are not necessarily unique, but every vertex, except the root vertex v_0 is uniquely defined by the tuple (v, t, i) , where v is the parent vertex (that is either the root vertex, or another vertex defined by a tuple), $t \in \mathbb{N}$ is the iteration when this vertex was created and $i \in [1..\lambda]$ is the number of the vertex among the vertices with the same v and t . The tree $T_0 = (V_0, E_0)$ at time $t = 0$ consists of the single (root) vertex v_0 that is labeled with the bit-string $c(v_0) = x_0$. Hence $E_0 = \emptyset$. If $T_t = (V_t, E_t)$ is defined for some $t \geq 0$, then we define the tree $T_{t+1} = (V_{t+1}, E_{t+1})$ as follows. For each vertex in V_t , we add λ vertices, connect them to this vertex, and generate their labels via standard-bit mutation from the parent. More precisely, let $N_{t+1} := \{(v_t, t+1, i) \mid v_t \in V_t, i \in [1..\lambda]\}$ and $V_{t+1} = V_t \cup N_{t+1}$. We call v_t the parent of $(v_t, t+1, i)$ and (v_t, t, i) the i -th child of v_t in iteration $t+1$. We generate the label $c(v_t, t+1, i)$ by applying standard-bit mutation to $c(v_t)$. We connect each new vertex with its parent, that is, we define $E_{t+1} = E_t \cup \{(v_t, (v_t, t+1, i)) \mid v_t \in V_t, i \in [1..\lambda]\}$. A simple example of a complete tree structure is shown in Figure 1.

It is easy to see that a complete tree at time t contains exactly $(\lambda + 1)^t$ nodes, since each vertex from V_t has exactly λ new children in V_{t+1} . As said

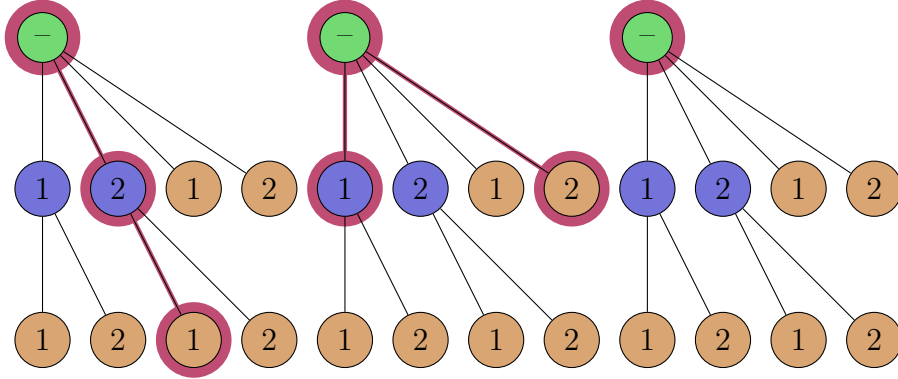


Figure 1: The structure of the complete tree for the $(3 + 2)$ EA after $t = 2$ iterations. The green vertices are the initial vertices, the blue vertices were created in the first iteration and the orange vertices were created in the second iteration. Each vertex is uniquely defined by a tuple of its parent vertex v , iteration it was created t and its number i among the children of its parent vertex created at the same iteration (the vertices in the figure are labeled with this number i). The highlighted vertices are the ones which were actually created by the algorithm. The labels are omitted in this illustration for reasons of readability

earlier, it thus massively overestimates the size of the true population of the EA.

For our purposes, it is not so much the total size of the tree that is important, but rather the number of nodes in a certain distance from the root. We estimate these in the following elementary lemma. Here and in the remainder, by *distance* we mean the graph theoretic distance, that is, the length of the (in this case unique) path between the two vertices. Observe that this can be different from the iteration in which a node was generated. For example, the vertex (v_0, t, i) , which is generated in iteration t from the initial vertex, has distance one from v_0 .

Lemma 10. *Let T_t be a complete tree at time t . Let $\ell \in \mathbb{N}_0$. Then T_t contains exactly*

$$\binom{t}{\ell} \lambda^\ell$$

nodes in distance exactly ℓ from the root.

Proof. If $t < \ell$, then there are no vertices in distance ℓ (recall that in our notation $\binom{t}{\ell} = 0$ in this case). Otherwise, let v be a vertex in distance exactly ℓ from the root. Then there are times $1 \leq t_1 < \dots < t_\ell \leq t$ and

offspring numbers $i_1, \dots, i_\ell \in [1..\lambda]$ such that with the recursive definition of the vertices v_1, \dots, v_ℓ via $v_d = (v_{d-1}, t_d, i_d)$ for all $d \in [1..\ell]$, we have $v = v_\ell$. Hence, there are at most $\binom{\ell}{t} \lambda^\ell$ vertices in distance ℓ from the root. Conversely, each tuple of times and offspring numbers as above defines a different vertex in distance ℓ . Hence, there are at least $\binom{\ell}{t} \lambda^\ell$ different vertices in distance ℓ from the root. \square

Since there is no selection in the complete tree, the vertex labels simply arise from repeated mutation. More precisely, a vertex in distance ℓ from the root has a label that is obtained from ℓ times applying mutation to the root label. This elementary observation allows to estimate the probability that a node label is equal to some target string.

Lemma 11. *Consider a complete tree with root label $c(v_0) = x_0$. Let $x^* \in \{0, 1\}^n$ with $H(x^*, x_0) \geq n/4$ (where H is the Hamming distance). Let x be the node label of a node in distance ℓ from v_0 . Then*

$$\Pr[x = x^*] \leq \min \left\{ 1, \left(\frac{\ell}{n-1} \right)^{n/4} \right\} =: p(\ell, n).$$

Proof. The probability that $x = x^*$ is at most the probability that each of the $H(x_0, x^*)$ bits in which x_0 and x^* differ was flipped in at least one of the ℓ applications of the mutation operator which generated x from x_0 . For one particular position the probability that this position was involved in one of ℓ mutations is $1 - (1 - \frac{1}{n})^\ell$. For $H(x_0, x^*)$ positions the probability that all of them were involved in one of ℓ mutations is

$$\left(1 - \left(1 - \frac{1}{n} \right)^\ell \right)^{H(x_0, x^*)} \leq \left(1 - \exp \left(-\frac{\ell}{n-1} \right) \right)^{\frac{n}{4}} \leq \left(\frac{\ell}{n-1} \right)^{\frac{n}{4}},$$

where we used the estimates $(1 - 1/n)^{(n-1)r} \geq e^{-r}$ valid for all $n \geq 1$ and any positive $r \in \mathbb{R}$, and $e^{-r} \geq 1 - r$ valid for all $r \in \mathbb{R}$. \square

We are now ready to prove our lower bound. Since the proof is valid not only for the ONEMAX function, but for any pseudo-Boolean function with a unique optimum, we formulate the result for such functions. We show extensions to many functions with multiple optima in the following section.

Theorem 4. *If μ is polynomial in n , then the $(\mu + \lambda)$ EA with any type of selection of the new parent population (including only selecting from the offspring population) needs an expected number of*

$$\Omega \left(\frac{n \log n}{\lambda} + \frac{\mu n}{\lambda} \right)$$

iterations to optimize any pseudo-Boolean function with a unique optimum.

If further $\frac{\lambda}{\mu} \geq e^e$, then the stronger bound

$$\Omega\left(\frac{n \log n}{\lambda} + \frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right)$$

holds.

Proof. Without any loss of generality in this proof we assume that the function optimized by the algorithm has an optimum in $x^* = (1, \dots, 1)$.

In our proofs we use the following tool. To prove that the expected runtime of the algorithm is $\Omega(f(n))$ for some function $f(n)$, it is enough to prove that the probability that the runtime is less than $f(n)$ is less than some constant $\gamma < 1$, since in this case the expected runtime is not less than $(1 - \gamma)f(n)$.

We first note that the bound $\Omega(\frac{n \log n}{\lambda})$ is easy to prove for the ONEMAX function. A short, but deep argument for this bound is that the $(\mu + \lambda)$ EA is an unary unbiased black-box complexity algorithm in the sense of Lehre and Witt [LW12]. Any such algorithm needs an expected number of $\Omega(n \log n)$ [LW12] or, more precisely, of at least $en \ln(n) - O(n)$ [DDY16] fitness evaluations to find the optimum of the ONEMAX function.

However, we prove the lower bounds for any function with a unique optimum, so we use an elementary argument essentially identical to the one of [Wit06] as follows. The lower bound $\Omega(\frac{n \log n}{\lambda})$ needs to be shown only in the case $\mu \leq c \log n$, where c is an arbitrarily small constant. For any bit position we have the probability q_1 that all individuals in the initial population have a zero-bit in that position that is calculated as

$$q_1 = \left(\frac{1}{2}\right)^\mu \geq \left(\frac{1}{2}\right)^{c \log(n)} = \exp(c \log(n) \log(1/2)) = n^{-c \log(2)}.$$

Thus, the expected number z of the bit positions such that all individuals in the initial population have a zero-bit in that position is

$$E[z] = \sum_{i=1}^n q_1 = n^{1-c \log(2)}.$$

We call such positions *initially wrong positions*. Since the bit values in each bit position and each initial individual are independent, each position is initially wrong or not independently on other positions. Hence, by Chernoff

bounds (see, e.g., Theorem 1.10.5 in [Doe20]) the probability q_2 that we have at least $E[z]/2 = n^{1-c\log(2)}/2$ such bit positions is calculated as

$$q_2 = \Pr [z \geq (1 - \delta)E[z]] \geq 1 - \exp\left(-\frac{\delta^2 E[z]}{2}\right) = 1 - \exp\left(-\frac{n^{1-c\log(2)}}{8}\right).$$

Now we are ready to show that the algorithm does not flip at least one of the bits in the initially wrong positions in $t := \lfloor \frac{\alpha(n-1)\log(n)}{\lambda} \rfloor$ iterations, where α is a constant that will be defined later, with a high (at least $1 - o(1)$) probability. We calculate the probability q_3 that one particular bit is flipped at least once in t iterations (or in λt mutations) as

$$\begin{aligned} q_3 &= 1 - \left(1 - \frac{1}{n}\right)^{\lambda t} = 1 - \left(1 - \frac{1}{n}\right)^{(n-1)\frac{\lambda t}{(n-1)}} \\ &\leq 1 - \exp\left(-\frac{t\lambda}{(n-1)}\right) \leq 1 - e^{-\alpha\log(n)} = 1 - n^{-\alpha}. \end{aligned}$$

If we have at least $n^{1-c\log(2)}/2$ initially wrong positions, then the probability q_4 that all of them are flipped at least once in t iterations is

$$\begin{aligned} q_4 &= q_3^{\frac{n^{1-c\log(2)}}{2}} \leq (1 - n^{-\alpha})^{\frac{n^{1-c\log(2)}}{2}} \\ &= (1 - n^{-\alpha})^{n^{\alpha \cdot \frac{n^{1-c\log(2)-\alpha}}{2}}} \leq \exp\left(-\frac{n^{1-c\log(2)-\alpha}}{2}\right) \end{aligned}$$

Thus we have the probability q_5 that at least one of the initially wrong bits is not flipped (and thus, the optimum is not found) in $t = \Theta(\frac{n\log(n)}{\lambda})$ iterations at least

$$\begin{aligned} q_5 &\geq q_2(1 - q_4) \\ &\geq \left(1 - \exp\left(-\frac{n^{1-c\log(2)}}{8}\right)\right) \left(1 - \exp\left(-\frac{n^{1-c\log(2)-\alpha}}{2}\right)\right) \quad (13) \\ &\geq 1 - 2 \exp\left(-\frac{n^{1-c\log(2)-\alpha}}{8}\right). \end{aligned}$$

Hence, if α and c satisfy $c\log(2) + \alpha < 1$, (e.g., $\alpha := \frac{1}{2}$ and $c := \frac{1}{2}$) then the expected runtime of the algorithm is $\Omega(\frac{n\log(n)}{\lambda})$.

To prove the remaining two bounds, we argue as follows. Again using a simple Chernoff bound argument, we first observe that the probability q_6 that the number of zero-bits y in the one particular individual in the initial population is less than $n/4$, is estimated as

$$q_6 = \Pr \left[y \leq \frac{E[y]}{2} \right] = \Pr [y \leq (1 - 1/2)E[y]] \leq \exp\left(-\frac{n}{16}\right).$$

Hence, all μ individuals of the initial population have a Hamming distance of at least $n/4$ from the optimum x^* with probability

$$q_7 = (1 - q_6)^\mu \geq \left(1 - \exp\left(-\frac{n}{16}\right)\right)^\mu \geq \exp\left(-\frac{\mu}{e^{\frac{n}{16}} - 1}\right)$$

Since μ is polynomial in n , we have $\frac{\mu}{e^{\frac{n}{16}} - 1} = o(1)$ and therefore, $q_7 = 1 - o(1)$. Further in this proof we assume that all initial individuals have at least $n/4$ zero-bits.

Clearly, a run of the $(\mu + \lambda)$ EA creates a subforest of μ disjoint complete trees with random root labels (complete forest). Whether a node of the complete forest appears in the forest describing the run of the $(\mu + \lambda)$ EA (the forest of the family trees) depends on the node labels (more precisely, on their fitness). However, regardless of the node labels the following is true: If some node v_s is present in the population at iteration t , then the edge $(v_s, (v_s, t, i))$ is present in the subforest at most with probability $1/\mu$, because for this it is necessary that the i -th offspring generated in iteration t chooses v_s as parent. Consequently, regardless of the nodes labels, the probability that a node in distance ℓ from the root in the complete forest enters the population of the $(\mu + \lambda)$ EA, is at most $\mu^{-\ell}$. Since we have not taken into account the node labels, we observe that the probability that a particular node of the complete forest (i) is labeled with the optimum and (ii) makes it into the population of the $(\mu + \lambda)$ EA, is at most $\mu^{-\ell}p(\ell, n)$ with $p(\ell, n)$ as defined in Lemma 11.

Using a union bound over all nodes in the complete forest up to iteration t , cf. Lemma 10, we see that the probability that the $(\mu + \lambda)$ EA finds the optimum within t iterations, is at most

$$q_{opt} \leq \mu \sum_{\ell=0}^t \binom{t}{\ell} \left(\frac{\lambda}{\mu}\right)^\ell p(\ell, n). \quad (14)$$

Let first $t := \lfloor \mu n / 8e\lambda \rfloor$. Using the inequality $\binom{t}{\ell} \leq (et/\ell)^\ell$ that follows from Stirling's formula, we estimate the summand $s(\ell) := \binom{t}{\ell} \left(\frac{\lambda}{\mu}\right)^\ell p(\ell, n)$ of q_{opt} for every $\ell \in [0..t]$.

- By Lemma 11 we have $p(\ell, n) \leq 1$. Thus, if $\ell \geq n/4$, we estimate

$$s(\ell) = \binom{t}{\ell} \left(\frac{\lambda}{\mu}\right)^\ell p(\ell, n) \leq \left(\frac{et\lambda}{\ell\mu}\right)^\ell \leq \left(\frac{n}{8\ell}\right)^\ell \leq (1/2)^\ell \leq (1/2)^{n/4}.$$

- By Lemma 11 we have $p(\ell, n) \leq (\ell/(n-1))^{n/4}$. Hence, if $n/4 \geq \ell > 0$, we estimate

$$s(\ell) = \binom{t}{\ell} \left(\frac{\lambda}{\mu}\right)^\ell p(\ell, n) \leq \left(\frac{et\lambda}{\ell\mu}\right)^\ell \left(\frac{\ell}{n-1}\right)^{n/4} \leq \left(\frac{n}{8\ell}\right)^\ell \left(\frac{\ell}{n-1}\right)^{n/4}$$

$$\leq \left(\frac{n}{4\ell}\right)^\ell \left(\frac{\ell}{n-1}\right)^{n/4} \leq \left(\frac{n}{4\ell} \cdot \frac{\ell}{n-1}\right)^{n/4} \leq (1/2)^{n/4}.$$

- Finally, for $\ell = 0$ we have $p(\ell, n) = 0$ and thus $s(\ell) = 0$.

Consequently, the optimum is found in less than t iterations if either there is an individual with less than $n/4$ zero-bits in the initial population, or with an exponentially small probability otherwise. Therefore, the probability q_8 of finding the optimum in less than t iterations is bounded as

$$\begin{aligned} q_8 &\leq (1 - q_7) + q_7 \mu \sum_{\ell=0}^t s(\ell) \\ &\leq \left(1 - \exp\left(-\frac{\mu}{e^{\frac{n}{16}} - 1}\right)\right) + \mu \sum_{\ell=1}^t (1/2)^{n/4} \\ &\leq \frac{\mu}{e^{\frac{n}{16}} - 1} + \frac{\mu^2 n}{8e\lambda} (1/2)^{n/4} = o(1), \end{aligned} \tag{15}$$

since we assumed μ to be at most polynomial in n .

We finish the proof by showing the lower bound $\Omega\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right)$ in case when $\frac{\lambda}{\mu} \geq e^e$. For this purpose let $t = \lfloor \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1) \ln \frac{\lambda}{\mu}} \rfloor$. Using the complete tree notation we show that the probability that the algorithm finds an optimum in less than t iterations is very small.

For all $\ell \in [0..t]$ consider $s(\ell)$. Using the inequality $\binom{t}{\ell} \leq (et/\ell)^\ell$ we estimate the upper bound for it as follows.

$$\begin{aligned} s(\ell) &= \binom{t}{\ell} \left(\frac{\lambda}{\mu}\right)^\ell p(\ell, n) \leq \left(\frac{et\lambda}{\ell\mu}\right)^\ell \left(\frac{\ell}{n-1}\right)^{n/4} \\ &= \exp\left(\ell \ln \frac{et\lambda}{\ell\mu} + \frac{n}{4} \ln \frac{\ell}{n-1}\right). \end{aligned} \tag{16}$$

Consider precisely the argument of the exponential function from the last equality in (16). For this purpose define $f(\ell) := \ell \ln \frac{et\lambda}{\ell\mu} + \frac{n}{4} \ln \frac{\ell}{n-1}$. By considering the derivative of $f(\ell)$ on segment $[0, t]$ one can see that it is a monotonically increasing function. Since $t \geq \ell$ and $\frac{\lambda}{\mu} \geq e^e$, we have $\frac{et\lambda}{\ell\mu} \geq e^{e+1}$ and thus, $\ln \frac{et\lambda}{\ell\mu} \geq e + 1$. Hence,

$$f'(\ell) = \ln \frac{et\lambda}{\ell\mu} - 1 + \frac{n}{4\ell} \geq e + 1 - 1 > 0$$

Thus, $f(\ell)$ reaches its maximum when $\ell = t$. Therefore,

$$\begin{aligned}
f(\ell) &\leq f(t) \leq t \ln \frac{e\lambda}{\mu} + \frac{n}{4} \ln \frac{t}{n-1} \\
&\leq \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1) \ln \frac{\lambda}{\mu}} \left(\ln \frac{\lambda}{\mu} + 1 \right) + \frac{n}{4} \ln \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1)(n-1) \ln \frac{\lambda}{\mu}} \\
&= \frac{(e-2)}{4(e+1)} n \ln \ln \frac{\lambda}{\mu} \left(1 + \frac{1}{\ln \frac{\lambda}{\mu}} \right) \\
&\quad + \frac{n}{4} \left(\ln \ln \ln \frac{\lambda}{\mu} - \ln \ln \frac{\lambda}{\mu} + \ln \frac{(e-2)n}{4(e+1)(n-1)} \right) \\
&\leq \frac{n}{4} \ln \ln \frac{\lambda}{\mu} \left(\frac{(e-2)}{(e+1)} \left(1 + \frac{1}{e} \right) + \frac{\ln \ln \ln \frac{\lambda}{\mu}}{\ln \ln \frac{\lambda}{\mu}} - 1 + \frac{\ln \frac{(e-2)n}{4(e+1)(n-1)}}{\ln \ln \frac{\lambda}{\mu}} \right).
\end{aligned}$$

Notice that $\frac{\ln x}{x} \leq \frac{1}{e}$ for all $x \geq 1$ and that $\ln \frac{(e-2)n}{4(e+1)(n-1)} < 0$ for all $n > 1$. Therefore we have

$$f(\ell) \leq \frac{n}{4} \ln \ln \frac{\lambda}{\mu} \left(\frac{(e-2)}{(e+1)} \left(1 + \frac{1}{e} \right) - \left(1 - \frac{1}{e} \right) \right) = -\frac{n \ln \ln \frac{\lambda}{\mu}}{4e}.$$

Thus, by (16) we have

$$s(\ell) \leq \exp \left(-\frac{n \ln \ln \frac{\lambda}{\mu}}{4e} \right) = \left(\ln \frac{\lambda}{\mu} \right)^{-n/4e}.$$

By (14) summing up $\mu s(\ell)$ for all $\ell \in [0..t]$ we obtain the following upper bound on the probability q_9 that the algorithm finds the optimum in less than $t = \Theta\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right)$ iterations.

$$\begin{aligned}
q_9 &\leq (1 - q_7) + q_7 \mu \sum_{\ell=0}^t s(\ell) \\
&\leq \frac{\mu}{e^{\frac{n}{16}} - 1} + \mu \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1) \ln \frac{\lambda}{\mu}} \left(\ln \frac{\lambda}{\mu} \right)^{-n/4e}.
\end{aligned} \tag{17}$$

Notice that q_9 is $o(1)$, since we assumed that μ is polynomial in n . Hence, the expected runtime of the algorithm is $\Omega\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right)$ \square

Comparison With Other Lower Bounds

Since all results involved are asymptotically tight, our lower bounds subsume the previous bounds for the $(\mu + 1)$ EA and the $(1 + \lambda)$ EA in the way as discussed for upper bounds in Section 3.4.

For general values of μ and λ , the only result [QYZ16] we are aware of proves that for any μ and λ that are at most polynomial in n the runtime of the $(\mu + \lambda)$ EA on every pseudo-boolean function with a unique global optimum is

$$\Omega\left(\frac{n \log n}{\lambda} + \frac{\mu}{\lambda} + \frac{n \log \log n}{\log n}\right). \quad (18)$$

By comparing the three terms of this bound with the corresponding terms of our bound

$$\Omega\left(\frac{n \log n}{\lambda} + \frac{n}{\lambda/\mu} + \frac{n \log^+ \log^+(\lambda/\mu)}{\log^+(\lambda/\mu)}\right),$$

we immediately see that our bound is asymptotically at least as large as the one in (18); note that for the third term, this follows trivially from the assumption that λ is polynomial in n and the fact that $x \mapsto \frac{\log \log(x)}{\log(x)}$ is decreasing for x sufficiently large.

There are two cases when our bound is asymptotically greater than (18). **Setting 1.** Let $\frac{\lambda}{\mu} = O(1)$ and $\mu = \omega(\log(n))$. Then our bound is $\Omega(\frac{n\mu}{\lambda})$, which is at least $\Omega(n)$. On the other hand, (18) is

$$\frac{n \log n}{\lambda} + \frac{\mu}{\lambda} + \frac{n \log \log n}{\log n} = \frac{n o(\mu)}{\lambda} + \frac{\mu}{\lambda} + o(n) = o\left(\frac{n\mu}{\lambda}\right).$$

Setting 2. Let $\log \frac{\lambda}{\mu} = \omega(\log n)$. This implies that $\frac{\lambda}{\mu} = \omega(n)$ and thus

$$\log n = o\left(\frac{n \log \log n}{\log n}\right) = o\left(\frac{\frac{\lambda}{\mu} \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right).$$

Therefore, we have

$$\frac{n \log n}{\lambda} = o\left(\frac{n \log \log \frac{\lambda}{\mu}}{\mu \log \frac{\lambda}{\mu}}\right) = o\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right).$$

Hence, the lower bound given in Theorem 4 simplifies to $\Omega(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}})$.

On the other hand, the bound (18) is of the asymptotically smaller order $o(\log n) + o(1) + O(\frac{n \log \log n}{\log n}) = O(\frac{n \log \log n}{\log n})$.

5 Extending the Lower Bounds to All Functions Having Not Excessively Many Global Optima

Since the family tree technique depends little on the particular function to be optimized, Witt [Wit06] extended his lower bounds for ONEMAX to a much broader class of functions. He proved that the $(\mu + 1)$ EA needs $\Omega(\mu n)$ iterations to find a global optimum of any function that satisfies one of the following conditions. (i) The function has at most $2^{o(n)}$ optima. (ii) All optima have at least $n/2 + \varepsilon n$ one-bits or all optima have at least $n/2 + \varepsilon n$ zero-bits, where $\varepsilon > 0$ is an arbitrary constant.

In this section we extend our lower bounds of Section 4 to a wide class of functions as well. In particular, we show that Witt's results are valid for all functions with at most $2^{\beta n}$ optima, where β is some constant less than $\frac{1}{16 \ln 2}$, regardless of the positions of the optima.

To reach our goal we exploit the fact that in Theorem 4 we proved very small values for the probabilities that the runtime is less than some threshold (see (13), (15) and (17)), while it would have been enough to prove that they are some constants less than one.

Theorem 5. *For any constant $\varepsilon > 0$ there exists another constant $c > 0$ such that if $\mu < c \ln n$, then for any n -dimensional pseudo-Boolean function with not more than $2^{n^{1-\varepsilon}}$ optima the $(\mu + \lambda)$ EA takes at least $\Omega(\frac{n \log n}{\lambda})$ iterations in expectation and with high probability to find an optimum.*

Proof. Let c be some arbitrary small positive constant and let $\mu < c \ln n$. By (13) the probability that the algorithm finds a particular optimum in less than $t := \frac{\alpha n \log n}{\lambda}$ iterations (where α is some arbitrary constant) is

$$1 - q_5 \leq 2 \exp\left(-\frac{n^{1-c \ln 2 - \alpha}}{8}\right).$$

If we have at most $2^{n^{1-\varepsilon}}$ optima, then by a union bound over all optima we obtain that the probability q_{10} that the algorithm finds an optimum in less than t iterations is

$$\begin{aligned} q_{10} &\leq (1 - q_5) 2^{n^{1-\varepsilon}} \leq 2 \exp\left(-\frac{n^{1-c \ln 2 - \alpha}}{8}\right) \exp(n^{1-\varepsilon} \ln 2) \\ &= 2 \exp\left(n^{1-\varepsilon} \ln 2 - \frac{n^{1-c \ln 2 - \alpha}}{8}\right). \end{aligned}$$

This probability q_{10} tends to zero with growing n if and only if the argument of the exponential function tends to negative infinity. It does so if and only if α and c satisfy $\alpha + c \ln 2 < \varepsilon$. Since ε is a positive constant, we can choose $\alpha := \varepsilon/2$ and $c := \varepsilon/2$ to satisfy this condition. \square

The actual reason that the algorithm cannot find an optimum faster than in $\Omega(\frac{n \log n}{\lambda})$ iterations is the coupon collector effect when the algorithm tries to flip the few wrong bits left in the end of the optimization. However, if we have $2^{\Theta(n)}$ optima, the algorithm avoids this effect. To illustrate this idea consider the $(1+1)$ EA that optimizes the ONEMAX function, but the bit-strings with less than cn zero-bits, where c is some small constant, are considered optimal. Thus, this function has no more than $O(2^{c \log_2(1/c)n}) \subseteq 2^{\Theta(n)}$ optima. Clearly, the runtime of the $(1+1)$ EA on such function is linear, which may be proven with simple additive drift argument.

The following two theorems extend our $\Omega(\frac{n\mu}{\lambda})$ and $\Omega(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}})$ bounds to the functions with $2^{O(n)}$ optima.

Theorem 6. *If μ is at most polynomial in n , then the $(\mu + \lambda)$ EA optimizes any pseudo-Boolean function with at most $2^{\beta n}$ optima, where β is some constant less than $\frac{1}{16 \ln 2}$, in $\Omega(\frac{\mu n}{\lambda})$ iterations. If $\frac{\lambda}{\mu} > e^e$, then the stronger bound $\Omega(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}})$ holds.*

Proof. By (15) the probability that the algorithm finds a particular optimum in less than $t := \lfloor \frac{\mu n}{8e\lambda} \rfloor$ iterations is

$$q_8 \leq \frac{\mu}{e^{\frac{n}{16}} - 1} + \frac{\mu^2 n}{8e\lambda} \left(\frac{1}{2}\right)^{\frac{n}{4}}.$$

By a union bound taken over no more than $2^{\beta n}$ optima, the probability q_{11} that the algorithm finds any optimum in this time is

$$q_{11} \leq q_8 2^{\beta n} \leq \frac{\mu e^{(\ln 2)\beta n - \frac{n}{16}}}{1 - e^{-\frac{n}{16}}} + \frac{\mu^2 n}{8e\lambda} 2^{\beta n - \frac{n}{4}}.$$

Since $\beta < \frac{1}{16 \ln 2}$ and β is a constant, we have both $(\ln 2)\beta n - \frac{n}{16} < 0$ and $\beta n - \frac{n}{4} < 0$ (and both of them are linear in n). Thus, q_{11} tends to zero with growing n . Hence, the expected runtime of the algorithm is $\Omega(t) = \Omega(\frac{\mu n}{\lambda})$.

To prove the $\Omega(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}})$ bound we argue in a similar way. By (17) the probability that the algorithm finds a particular optimum in less than

$t := \lfloor \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1) \ln \frac{\lambda}{\mu}} \rfloor$ iterations is

$$q_9 \leq \frac{\mu}{e^{\frac{n}{16}} - 1} + \mu \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1) \ln \frac{\lambda}{\mu}} \left(\ln \frac{\lambda}{\mu} \right)^{-n/4e}.$$

By a union bound taken over no more than $2^{\beta n}$ optima, the probability q_{12} that the algorithm finds any optimum in this time is

$$q_{12} \leq q_9 2^{\beta n} \leq \frac{\mu e^{\beta n \ln 2 - n/16}}{1 - e^{-\frac{n}{16}}} + \mu \frac{(e-2)n \ln \ln \frac{\lambda}{\mu}}{4(e+1) \ln \frac{\lambda}{\mu}} e^{\beta n \ln 2 - n/4}.$$

Since $\beta < \frac{1}{16 \ln 2}$ and β is a constant, we have both $(\ln 2)\beta n - \frac{n}{16} < 0$ and $\beta n \ln 2 - \frac{n}{4} < 0$ (and both of them are linear in n). Thus, q_{12} tends to zero with growing n . Hence, the expected runtime of the algorithm is $\Omega(t) = \Omega\left(\frac{n \log \log \frac{\lambda}{\mu}}{\log \frac{\lambda}{\mu}}\right)$. \square

6 Analysis of the $(\lambda \overset{1:1}{+} \lambda)$ EA

In this section we prove that our results (both upper bound from Theorem 2 and lower bound from Theorem 4) hold in an analogous fashion also for the $(\lambda \overset{1:1}{+} \lambda)$ EA, that is, we show that this algorithm optimizes ONEMAX in an expected number of $\Theta\left(\frac{n \log n}{\lambda} + n\right)$ iterations. This improves over the $O\left(\frac{n \log n}{\lambda} + n \log \lambda\right)$ proven bound and the $O\left(\frac{n \log n}{\lambda} + n \log \log n\right)$ conjecture of [CHS⁺09].

Due to the differences in the algorithms, to prove our results we obviously cannot just apply the previous theorems in this work to the case $\lambda = \mu$. We recall that the $(\lambda \overset{1:1}{+} \lambda)$ EA uses a different parent selection. While the classic $(\mu + \lambda)$ EA chooses each parent independently and uniformly at random from the μ individuals, the $(\lambda \overset{1:1}{+} \lambda)$ EA creates exactly one offspring from each parent. The pseudocode of the $(\lambda \overset{1:1}{+} \lambda)$ EA is shown in Algorithm 2. We note that [CHS⁺09] also use a slightly different way of selecting the next parent population. In principle, they take as new parent population the μ best individuals among parents and offspring (plus-selection). If this would lead to a new parent population only consisting of offspring, they remove the weakest offspring and replace it with the strongest individual from the previous parent population. Since this appears to be a not very common way of selecting the new population, we shall work with the classic plus-selection, favoring offspring in case of ties, and breaking further ties randomly (though, indeed,

Algorithm 2: The $(\lambda \stackrel{1:1}{+} \lambda)$ EA, maximizing a given function $f : \{0, 1\}^n \rightarrow \mathbb{R}$, with population size λ and mutation rate p .

1 **Initialization:**
2 Create a population of λ individuals by choosing $x^{(i)} \in \{0, 1\}^n$,
 $1 \leq i \leq \lambda$ uniformly at random. Let the multiset
 $X^{(0)} := \{x^{(1)}, \dots, x^{(\lambda)}\}$ be the population at time 0. Let $t := 0$.
3 **Optimization:**
4 **while** *an optimum has not been reached* **do**
5 $X' := X^{(t)}$;
6 **Mutation phase:**
7 **for** $i = 1, \dots, \lambda$ **do**
8 $x :=$ the i -th individual from $X^{(t)}$ (deterministic selection);
9 Create x' by flipping each bit of x with probability p ;
10 $X' := X' \cup \{x'\}$;
11 **Selection phase:**
12 Create the multiset $X^{(t+1)}$, the population at time $t + 1$, by
deleting the λ individuals with lowest f -value in X' ;
13 $t := t + 1$;

the tie-breaking is not important when optimizing ONEMAX via unary unbiased black-box algorithms). We note without proof that the following results and proofs are valid for the precise algorithm regarded in [CHS⁺09] as well.

We start by proving the upper bound for the runtime.

Theorem 7. *The expected runtime of the $(\lambda \stackrel{1:1}{+} \lambda)$ EA on the ONEMAX function is $O(\frac{n \log n}{\lambda} + n)$.*

Proof. We aim at adapting Theorem 2 for the $(\lambda \stackrel{1:1}{+} \lambda)$ EA. For this purpose we note that the proof of Theorem 2 only depends on the expected level improvement times $E[\tilde{T}_i]$ computed in Corollary 1, which again depend on the times needed for increasing the number of fit individuals computed in Lemma 3. Therefore, it suffices to show that the estimates of Lemma 3 and Corollary 1 are also valid for the $(\lambda \stackrel{1:1}{+} \lambda)$ EA.

We prove that Lemma 3 holds for the $(\lambda \stackrel{1:1}{+} \lambda)$ EA by observing that the probability $p_2(j)$ to create at least one copy of the fit individual satisfies the same estimate as the one used for the $(\mu + \lambda)$ EA, which is (3), with $\mu = \lambda$. For the $(\lambda \stackrel{1:1}{+} \lambda)$ EA, $p_2(j)$ is at least the probability that at least one of the j

fit parent individuals creates as offspring a copy of it. By Lemma 2 we have

$$p_2(j) \geq 1 - \left(1 - \left(1 - \frac{1}{n}\right)^n\right)^j \geq 1 - \left(1 - \frac{1}{2e}\right)^j \geq \frac{1}{1 + \frac{2e}{j}},$$

which is the same estimate as for the $(\mu + \lambda)$ EA (with $\mu = \lambda$).

To prove that Corollary 1 holds for the $(\lambda \overset{1:1}{+} \lambda)$ EA as well, it is sufficient to show that the probability $p''(i)$ to create a superior individual satisfies as well the estimate (5) in the case $\mu = \lambda$. The probability $p''(i)$ is at least the probability that for at least one of the $\mu_0(i)$ best individuals the offspring is better than its parent. Using Lemma 2 we calculate

$$\begin{aligned} p''(i) &\geq 1 - \left(1 - \frac{n-i}{n} \left(1 - \frac{1}{n}\right)^{n-1}\right)^{\mu_0(i)} \geq 1 - \left(1 - \frac{n-i}{en}\right)^{\mu_0(i)} \\ &\geq 1 - \frac{1}{1 + \frac{\mu_0(i)(n-i)}{ne}}, \end{aligned}$$

which is the same value as in Corollary 1 when $\mu = \lambda$. \square

Comparing this bound with the bound $O(\frac{n \log n}{\lambda} + n \log \lambda)$ proven in [CHS⁺09] and the bound $O(\frac{n \log n}{\lambda} + n \log \log n)$ conjectured in the same work, we immediately see that ours is at least as strong as these two for all values of λ . For $\lambda = \omega(\frac{\log n}{\log \log n})$, our bound is asymptotically smaller than both the proven bound and the conjecture.

We now prove a matching lower bound, which agrees with the one of Theorem 4 in the case of $\mu = \lambda$.

Theorem 8. *If λ is polynomial in n then the expected runtime of the $(\lambda \overset{1:1}{+} \lambda)$ EA on the ONEMAX function is $\Omega(\frac{n \log n}{\lambda} + n)$.*

Proof. We show that the main arguments of the proof for this bound in Theorem 4 are also valid for this parent selection mechanism.

To prove the $\Omega(\frac{n \log n}{\lambda})$ bound we can repeat the arguments from Theorem 4 without any changes. One needs to prove this bound only for $\lambda < c \log n$ for some arbitrary small constant c . The main argument is that with high probability there is a set of bits which were in a wrong position in all initial individuals and that at least one of those bits was not flipped by any of $t\lambda$ applications of the mutation operator for some $t = \Theta(\frac{n \log n}{\lambda})$. This argument stays valid for the fair parent selection as well.

To prove the $\Omega(n)$ bound we consider the complete trees for the $(\lambda \overset{1:1}{+} \lambda)$ EA. Since in a run of the $(\lambda \overset{1:1}{+} \lambda)$ EA each individual in the population creates exactly one offspring, the complete trees now have a slightly

different structure, namely each node of the tree has exactly one child at each time step (instead of λ children). In return, we cannot argue that each edge is present in the true family tree with probability at most $1/\mu$ only (so we assume that all these edges are in fact present). Since $\lambda = \mu$, these two effects cancel.

More precisely, following the proof of Theorem 4 we argue that with high probability $q_7 \geq \exp(-\frac{\lambda}{e^{\frac{n}{16}} - 1})$ all initial individuals have at least $n/4$ wrong bits. Next, we argue that in an analogous fashion as in (14) – and this is where the two effects truly cancel – the probability q_{opt} that the optimum occurs in any tree in less than $t := \lceil \frac{n}{8e} \rceil$ iterations is at most

$$q_{opt} \leq \lambda \sum_{\ell=0}^t \binom{t}{\ell} p(\ell, n) \leq \lambda t \left(\frac{1}{2}\right)^{n/4}.$$

Since we only consider λ that is polynomial in n , this entity tends to zero, when n tends to infinity. Therefore, the probability that the algorithm finds an optimum in $t = \Theta(n)$ iterations is at most $(1 - q_7) + q_7 q_{opt}$ that is less than some constant, if n is large enough. Hence, the expected runtime of the $(\lambda \overset{1:1}{+} \lambda)$ EA is $\Omega(n)$. \square

7 Discussion and Conclusion

In this work, we determined – tight apart from constant factors – the runtime of the $(\mu + \lambda)$ EA on the ONEMAX benchmark problem. This is thus one of the few tight runtime analyses taking into account more than a single parameter ([GW17, DD18] are the other two such works we are aware of).

Not surprisingly for a simple function like ONEMAX, our result does not indicate that it is advantageous to use larger parent or offspring populations. Indeed, it follows from [Wit13, Theorem 6.2] (see [Doe19] for a simplified proof) that for any μ and λ the runtime of the $(\mu + \lambda)$ EA stochastically dominates the runtime of the $(1 + 1)$ EA with best-of- μ initialization. The runtime difference between the $(1 + 1)$ EA with best-of- μ initialization and with the usual random initialization is small, roughly an additive $\Theta(\sqrt{n \ln \mu})$ term [dPdLDD15].

While our result does not show an advantage of using larger populations, it does show that using moderate-size populations is not overly costly. For example, as long as $\mu, \lambda = O(\log n)$, the $(\mu + \lambda)$ EA takes $\Theta(n \log n)$ fitness evaluations to find the optimum. This observation could indicate that using such population sizes is generally an interesting idea – we could speculate

that there is no harm from using such populations, but there could be other advantages.

In the light of recent other work, our work suggests two directions for further research. In [GW17], a precise runtime analysis for the $(1 + \lambda)$ EA with general mutation rate c/n , c a constant, on the ONEMAX benchmark was conducted. It suggests that the precise mutation rate is important when λ is small, but less decisive when λ is large. It would be interesting to know to what extent this result carries over to the $(\mu + \lambda)$ EA. In [BLS14, DGWY19, DWY18], it was shown that various dynamic choices of the mutation rate can reduce the runtime of the $(1 + \lambda)$ EA on ONEMAX. Again, it would be interesting to see to what extent a similar behavior is true for the $(\mu + \lambda)$ EA.

Acknowledgement

We are thankful to Jiefeng Fang and Tangi Hetet for their contributions to the preliminary version [ADFH18] of this work. The second author would like to thank Jon Rowe for pointing him to the arguments used in [RS14], which were used in the proof of Lemma 2. The first author was supported by the Government of Russian Federation (Grant 08-08).

References

- [ADFH18] Denis Antipov, Benjamin Doerr, Jiefeng Fang, and Tangi Hetet. Runtime analysis for the $(\mu + \lambda)$ EA optimizing OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 1459–1466. ACM, 2018.
- [ADY19] Denis Antipov, Benjamin Doerr, and Quentin Yang. The efficiency threshold for the offspring population size of the (μ, λ) EA. In *Genetic and Evolutionary Computation Conference, GECCO 2019*, pages 1461–1469. ACM, 2019.
- [BDN10] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Parallel Problem Solving from Nature, PPSN XI, Part I*, pages 1–10, 2010.
- [BLS14] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *Parallel Problem Solving from Nature, PPSN 2014*, pages 892–901. Springer, 2014.

- [CDF14] Sylvain Colin, Benjamin Doerr, and Gaspard Férey. Monotonic functions in EC: anything but monotone! In *Genetic and Evolutionary Computation Conference, GECCO 2014*, pages 753–760. ACM, 2014.
- [CHS⁺09] Tianshi Chen, Jun He, Guangzhong Sun, Guoliang Chen, and Xin Yao. A new approach for analyzing average time complexity of population-based evolutionary algorithms on unimodal problems. *IEEE Transactions on Systems, Man, and Cybernetics, Part B (Cybernetics)*, 39:1092–1106, 2009.
- [DD18] Benjamin Doerr and Carola Doerr. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709, 2018.
- [DDE15] Benjamin Doerr, Carola Doerr, and Franziska Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.
- [DDY16] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. In *Genetic and Evolutionary Computation Conference, GECCO 2016*, pages 1123–1130. ACM, 2016.
- [DFK⁺18] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Escaping local optima using crossover with emergent diversity. *IEEE Transactions on Evolutionary Computation*, 22:484–497, 2018.
- [DG13] Benjamin Doerr and Leslie A. Goldberg. Adaptive drift analysis. *Algorithmica*, 65:224–250, 2013.
- [DGWY19] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. The $(1 + \lambda)$ evolutionary algorithm with self-adjusting mutation rate. *Algorithmica*, 81:593–631, 2019.
- [DHK12] Benjamin Doerr, Edda Happ, and Christian Klein. Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science*, 425:17–33, 2012.
- [DJS⁺13] Benjamin Doerr, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges. Mutation rate matters even

- when optimizing monotone functions. *Evolutionary Computation*, 21:1–21, 2013.
- [DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the $(1+1)$ evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.
- [DJW12] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64:673–697, 2012.
- [DK15] Benjamin Doerr and Marvin Künnemann. Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm—different asymptotic runtimes for different instances. *Theoretical Computer Science*, 561:3–23, 2015.
- [DKV13] Benjamin Doerr, Bojana Kodric, and Marco Voigt. Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In *Congress on Evolutionary Computation, CEC 2013*, pages 432–439. IEEE, 2013.
- [DL16] Duc-Cuong Dang and Per Kristian Lehre. Runtime analysis of non-elitist populations: From classical optimisation to partial information. *Algorithmica*, 75:428–461, 2016.
- [DLMN17] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2017*. ACM, 2017.
- [Doe18] Benjamin Doerr. An elementary analysis of the probability that a binomial random variable exceeds its expectation. *Statistics and Probability Letters*, 139:67–74, 2018.
- [Doe19] Benjamin Doerr. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science*, 773:115–137, 2019.
- [Doe20] Benjamin Doerr. Probabilistic tools for the analysis of randomized optimization heuristics. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 1–87. Springer, 2020. Also available at <https://arxiv.org/abs/1801.06733>.

- [dPdLDD15] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. Money for nothing: Speeding up evolutionary algorithms through better initialization. In *Genetic and Evolutionary Computation Conference, GECCO 2015*, pages 815–822. ACM, 2015.
- [Dro05] Stefan Droste. Not all linear functions are equally difficult for the compact genetic algorithm. In *Genetic and Evolutionary Computation Conference, GECCO 2005*, pages 679–686. ACM, 2005.
- [DWH18] Benjamin Doerr, Carsten Witt, and Jing Yang. Runtime analysis for self-adaptive mutation rates. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 1475–1482. ACM, 2018.
- [GK16] Christian Gießen and Timo Kötzing. Robustness of populations in stochastic environments. *Algorithmica*, 75:462–489, 2016.
- [GM14] Spencer Greenberg and Mehryar Mohri. Tight lower bound on the probability of a binomial exceeding its expectation. *Statistics & Probability Letters*, 86:91–98, 2014.
- [GW17] Christian Gießen and Carsten Witt. The interplay of population size and mutation probability in the $(1 + \lambda)$ EA on OneMax. *Algorithmica*, 78:587–609, 2017.
- [HY01] Jun He and Xin Yao. Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:51–81, 2001.
- [HY04] Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3:21–35, 2004.
- [JJW05] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation*, 13:413–440, 2005.
- [JW05] Jens Jägersküpfer and Carsten Witt. Rigorous runtime analysis of a $(\mu + 1)$ ES for the sphere function. In *Genetic and Evolutionary Computation Conference, GECCO 2005*, pages 849–856. ACM, 2005.

- [JZ11] Thomas Jansen and Christine Zarges. On benefits and drawbacks of aging strategies for randomized search heuristics. *Theoretical Computer Science*, 412:543–559, 2011.
- [Leh10] Per Kristian Lehre. Negative drift in populations. In *Parallel Problem Solving from Nature, PPSN 2010*, pages 244–253. Springer, 2010.
- [Len18] Johannes Lengler. A general dichotomy of evolutionary algorithms on monotone functions. In *Parallel Problem Solving from Nature, PPSN 2018, Part II*, pages 3–15. Springer, 2018.
- [LW12] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.
- [LY12] Per Kristian Lehre and Xin Yao. On the impact of mutation-selection balance on the runtime of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 16:225–241, 2012.
- [QYZ16] Chao Qian, Yang Yu, and Zhi-Hua Zhou. A lower bound analysis of population-based evolutionary algorithms for pseudo-boolean functions. In *International Conference on Intelligent Data Engineering and Automated Learning, IDEAL 2016*, pages 457–467. Springer, 2016.
- [Rob55] Herbert Robbins. A remark on Stirling’s formula. *The American Mathematical Monthly*, 62:26–29, 1955.
- [RRS98] Yuval Rabani, Yuri Rabinovich, and Alistair Sinclair. A computational view of population genetics. *Random Structures & Algorithms*, 12:313–334, 1998.
- [RS14] Jonathan E. Rowe and Dirk Sudholt. The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. *Theoretical Computer Science*, 545:20–38, 2014.
- [Sud09] Dirk Sudholt. The impact of parametrization in memetic evolutionary algorithms. *Theoretical Computer Science*, 410:2511–2528, 2009.
- [Weg01] Ingo Wegener. Theoretical aspects of evolutionary algorithms. In *International Colloquium on Automata, Languages, and Programming, ICALP 2001*, pages 64–78. Springer, 2001.

- [Wit03] Carsten Witt. Population size vs. runtime of a simple EA. In *Congress on Evolutionary Computation, CEC 2003*, pages 1996–2003. IEEE, 2003.
- [Wit06] Carsten Witt. Runtime analysis of the $(\mu + 1)$ EA on simple pseudo-Boolean functions. *Evolutionary Computation*, 14:65–86, 2006.
- [Wit08] Carsten Witt. Population size versus runtime of a simple evolutionary algorithm. *Theoretical Computer Science*, 403:104–120, 2008.
- [Wit13] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing*, 22:294–318, 2013.
- [YQZ15] Yang Yu, Chao Qian, and Zhi-Hua Zhou. Switch analysis for running time analysis of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 19:777–792, 2015.