The (1+) evolutionary algorithm with self-adjusting mutation rate

Doerr, Benjamin; Witt, Carsten; Gießen, Christian; Yang, Jing

Published in:
Proceedings of 2017 Genetic and Evolutionary Computation Conference

Link to article, DOI:
10.1145/3071178.3071279

Publication date:
2017

Document Version
Peer reviewed version

Link back to DTU Orbit

Citation (APA):
Doerr, B., Witt, C., Gießen, C., & Yang, J. (2017). The (1+) evolutionary algorithm with self-adjusting mutation rate. In *Proceedings of 2017 Genetic and Evolutionary Computation Conference* (pp. 1351-1358). Association for Computing Machinery. https://doi.org/10.1145/3071178.3071279

The (1+λ) Evolutionary Algorithm with Self-Adjusting Mutation Rate[*]

Benjamin Doerr
Laboratoire d'Informatique (LIX)
École Polytechnique
Palaiseau, France

Christian Gießen
DTU Compute
Technical University of Denmark
Kgs. Lyngby, Denmark

Carsten Witt
DTU Compute
Technical University of Denmark
Kgs. Lyngby, Denmark

Jing Yang
Laboratoire d'Informatique (LIX)
École Polytechnique
Palaiseau, France

June 20, 2017

Abstract

We propose a new way to self-adjust the mutation rate in population-based evolutionary algorithms in discrete search spaces. Roughly speaking, it consists of creating half the offspring with a mutation rate that is twice the current mutation rate and the other half with half the current rate. The mutation rate is then updated to the rate used in that subpopulation which contains the best offspring.

We analyze how the $(1+\lambda)$ evolutionary algorithm with this self-adjusting mutation rate optimizes the OneMax test function. We prove that this dynamic version of the $(1+\lambda)$ EA finds the optimum in an expected optimization time (number of fitness evaluations) of $O(n\lambda/\log\lambda + n\log n)$. This time is asymptotically smaller than the optimization time of the classic $(1+\lambda)$ EA. Previous work shows that this performance is best-possible among all λ-parallel mutation-based unbiased black-box algorithms.

This result shows that the new way of adjusting the mutation rate can find optimal dynamic parameter values on the fly. Since our adjustment mechanism is simpler than the ones previously used for adjusting the mutation rate and does not have parameters itself, we are optimistic that it will find other applications.

1 Introduction

Evolutionary algorithms (EAs) have shown a remarkable performance in a broad range of applications. However, it has often been observed that this performance depends crucially on the use of the right parameter settings. Parameter optimization and parameter

[*] An extended abstract of this report will appear in the proceedings of the 2017 Genetic and Evolutionary Computation Conference (GECCO 2017).

control are therefore key topics in EA research. Since these have very different characteristics in discrete and continuous search spaces, we discuss in this work only evolutionary algorithms for discrete search spaces.

Theoretical research has contributed to our understanding of these algorithms with mathematically founded runtime analyses, many of which show how the runtime of an EA is determined by its parameters. The majority of these works investigate *static parameter settings*, i. e., the parameters are fixed before the start of the algorithm and are not changed during its execution. More recently, a number of results were shown which prove an advantage of *dynamic parameter settings*, that is, the parameters of the algorithm are changed during its execution. Many of these rely on making the parameters functionally dependent on the current state of the search process, e.g., on the fitness of the current-best individual. While this provably can lead to better performances, it leaves the algorithm designer with an even greater parameter setting task, namely inventing a suitable functional dependence instead of fixing numerical values for the parameters. This problem has been solved by theoretical means for a small number of easy benchmark problems, but it is highly unclear how to find such functional relations in the general case.

A more designer-friendly way to work with dynamic parameters is to modify the parameters based on simple rules taking into account the recent performance. A number of recent results shows that such *on the fly* or *self-adjusting* parameter settings can give an equally good performance as the optimal fitness-dependent parameter setting, however, with much less input from the algorithm designer. For example, good results have been obtained by increasing or decreasing a parameter depending on whether the current iteration improved the best-so-far solution or not, e.g., in a way resembling the 1/5-th rule from continuous optimization.

Such success-based self-adjusting parameter settings can work well when there is a simple monotonic relation between success and parameter value, e.g., when one speculates that increasing the size of the population in an EA helps when no progress was made. For parameters like the mutation rate, it is not clear what a success-based rule can look like, since a low success rate can either stem from a too small mutation rate (regenerating the parent with high probability) or a destructive too high mutation rate. In [17], a relatively complicated learning mechanism was presented that tries to learn the right mutation strength by computing a time-discounted average of the past performance stemming from different parameter values. This learning mechanism needed a careful trade-off between exploiting the currently most profitably mutation strength and experimenting with other parameter values and a careful choice of the parameter controlling by how much older experience is taken less into account than more recent observations.

1.1 A New Self-Adjusting Mechanism for Population-Based EAs

In this work, we propose an alternative way to adjust the mutation rate on the fly for algorithms using larger offspring populations. It aims at overcoming some of the difficulties of the learning mechanism just described. The simple idea is to create half the offspring with twice the current mutation rate and the other half using half the current rate. The mutation rate is then modified to the rate which was used to create the best of these offspring (choosing the winning offspring randomly among all best in case of ambiguity). We do not allow the mutation rate to leave the interval $[2/n, 1/4]$, so that the rates used in the subpopulations are always in the interval $[1/n, 1/2]$.

We add one modification to the very basic idea described in the first paragraph of this section. Instead of always modifying the mutation rate to the rate of the best offspring, we shall take this winner's rate only with probability a half and else modify the mutation rate to a random one of the two possible values (twice and half the current rate). Our motivation for this modification is that we feel that the additional random noise will not prevent the algorithm from adjusting the mutation rate into a direction that is more profitable. However, the increased amount of randomness may allow the algorithm to leave a possible basin of attraction of a locally optimal mutation rate. Observe that with probability $\Theta(1/n^2)$, a sequence of $\log_2 n$ random modification all in the same direction appears. Hence with this inverse-polynomial rate, the algorithm can jump from any mutation rate to any other (with the restriction that only a discrete set of mutation rates can appear). We note that the existence of random modifications is also exploited in our runtime analysis, which will show that the new self-adjusting mechanism selects mutation rates good enough to lead to the asymptotically optimal runtime among all dynamic choices of the mutation rate for the (1+λ) EA.

In this first work proposing this mechanism, we shall not spend much effort fine-tuning it, but rather show in a proof-of-concept manner that it can find very good mutation rates. In a real application, it is likely that better results are obtained by working with three subpopulations, namely an additional one using (that is, exploiting) the current mutation rate. Also, it seems natural that more modest adjustments of the mutation rate, that is, multiplying and dividing the rate by a number F that is smaller than the value $F = 2$ used by our mechanism, is profitable. We conduct some elementary experiments supporting this intuition in Section 8.

1.2 Runtime Analysis for the Self-Adjusting (1+λ) EA on OneMax

To prove that the self-adjusting mechanism just presented can indeed find good dynamic mutation rates, we analyse it in the purest possible setting, namely in the optimization of the classic test function

$$\textsc{OneMax} : \{0,1\}^n \to \mathbb{R}; (x_1, \ldots, x_n) \mapsto \sum_{i=1}^{n} x_i$$

via the (1+λ) EA (see Algorithm 1).

The runtime of the (1+λ) EA with fixed mutation rates on OneMax is well understood [11, 22]. In particular, Gießen and Witt [22] show that the expected runtime (number of generations) is $(1 \pm o(1)) \left(\frac{1}{2} \cdot \frac{n \ln \ln \lambda}{\ln \lambda} + \frac{e^r}{r} \cdot \frac{n \ln n}{\lambda}\right)$ when a mutation rate of r/n, r a constant, is used. Thus for λ not too large, the mutation rate determines the leading constant of the runtime, and a rate of $1/n$ gives the asymptotically best runtime.

As a consequence of their work on parallel black-box complexities, Badkobeh, Lehre, and Sudholt [1] showed that the (1+λ) EA with a suitable fitness-dependent mutation rate finds the optimum of OneMax in an asymptotically better runtime of $O(\frac{n}{\log\lambda} + \frac{n\log n}{\lambda})$, where the improvement is by a factor of $\Theta(\log\log\lambda)$. This runtime is best-possible among all λ-parallel unary unbiased black-box optimization algorithms. In particular, no other dynamic choice of the mutation rate in the (1+λ) EA can achieve an asymptotically better runtime. The way how the mutation rate depends on the fitness in the above result, however, is not trivial. When the parent individual has fitness distance d, then mutation rate employed is $p = \max\{\frac{\ln\lambda}{n\ln(en/d)}, \frac{1}{n}\}$.

Our main technical result is that the (1+λ) EA adjusting the mutation rate according to the mechanism described above has the same (optimal) asymptotic runtime. Consequently, the self-adjusting mechanism is able find on the fly a mutation rate that is sufficiently close to the one proposed in [1] to achieve asymptotically the same expected runtime.

Theorem 1. *Let $\lambda \geq 45$ and $\lambda = n^{O(1)}$. Let T denote the number of generations of the (1+λ) EA with self-adjusting mutation rate on* OneMax*. Then,*

$$E(T) = \Theta\left(\frac{n}{\log\lambda} + \frac{n\log n}{\lambda}\right).$$

This corresponds to an expected number of functions evaluations of $\Theta(\frac{\lambda n}{\log\lambda} + n\log n)$.

To the best of our knowledge, this is the first time that a simple mutation-based EA achieves a super-constant speed-up via a self-adjusting choice of the mutation rate.

As an interesting side remark, our proofs reveal that a quite non-standard but fixed mutation rate of $r = \ln(\lambda)/2$ also achieves the $\Theta(\log\log\lambda)$ improvement as it implies the bound of $\Theta(n/\log\lambda)$ generations if λ is not too small. Hence, the constant choice $r = O(1)$ as studied in [22] does not yield the asymptotically optimal number of generations unless λ is so small that the $n\log n$-term dominates.

Lemma 1. *Let $\lambda \geq 45$ and $\lambda = n^{O(1)}$, Let T denote the number of generations of the (1+λ) EA with fixed mutation rate $r = \ln(\lambda)/2$. Then,*

$$E(T) = O\left(\frac{n}{\log\lambda} + \frac{n\log n}{\sqrt{\lambda}}\right).$$

This corresponds to an expected number of functions evaluations of $O(\frac{\lambda n}{\log\lambda} + \sqrt{\lambda}n\log n)$.

The paper is structured as follows: In Section 2 we give a overview over previous analyses of the (1+λ) EA and of self-adjusting parameter control mechanism in EAs from a theoretical perspective. In Section 3 we give the algorithm and the mutation scheme. For convenience, we also state some key theorems that we will frequently use in the rest of the paper. The next three sections deal with the runtime analysis of the expected time spent by the (1+λ) EA on OneMax in each of three regions of the fitness distance d. We label these regions the *far region*, *middle region* and *near region*, each of which will be dealt with in a separate section. The proof of the main theorem and of Lemma 1 is then given in Section 7. Finally, we conclude in Section 9.

2 Related Work

Since this is a theoretically oriented work on how a dynamic parameter choice speeds up the runtime of the (1+λ) EA on the test function OneMax, let us briefly review what is known about the theory of this EA and dynamic parameter choices in general.

2.1 The (1+λ) EA

The first to conduct a rigorous runtime analysis of the (1+λ) EA were Jansen, De Jong, and Wegener [24]. They proved, among other results, that when optimizing OneMax a linear speed-up exists up to a population size of $\Theta(\log(n)\log\log(n)/\log\log\log(n))$, that is, for $\lambda = O(\log(n)\log\log(n)/\log\log\log(n))$, finding the optimal solution takes an expected number of $\Theta(n\log(n)/\lambda)$ generations, whereas for larger λ at least $\omega(n\log(n)/\lambda)$ generations are necessary. This picture was completed in [11] with a proof that the expected number of generations taken to find the optimum is $\Theta(\frac{n\log n}{\lambda} + \frac{n\log\log\lambda}{\log\lambda})$. The implicit constants were determined in [22], giving the bound of $(1 \pm o(1))(\frac{1}{2}\frac{n\ln\ln\lambda}{\ln\lambda} + \frac{e^r}{r}\frac{n\ln n}{\lambda})$, for any constant r, as mentioned in the introduction.

Aside from the optimization behavior on OneMax, not too much is known for the (1+λ) EA, or is at least not made explicit (it is easy to see that waiting times for an improvement which are larger than λ reduce by a factor of $\Theta(\lambda)$ compared to one-individual offspring populations). Results made explicit are the $\Theta(n^2/\log(n) + n)$ expected runtime (number of generations) on LeadingOnes [24], the worst-case $\Theta(n + n\log(n)/\lambda)$ expected runtime on linear functions [11], and the $O(m^2(\log n + \log w_{\max})/\lambda)$ runtime estimate for minimum spanning trees valid for $\lambda \leq m^2/n$ [31].

2.2 Dynamic Parameter Choices

While it is clear the EAs with parameters changing during the run of the algorithm (dynamic parameter settings) can be more powerful than those only using static parameter settings, only recently considerable advantages of dynamic choices could be demonstrated by mathematical means (for discrete optimization problems; in continuous optimization, step size adaptation is obviously necessary to approach arbitrarily closely a target point). To describe the different ways to dynamically control parameters, we use in the following the language proposed in Eiben, Hinterding, and Michalewicz [21] and its extension from [10].

2.2.1 Deterministic Parameter Control

In this language, *deterministic parameter control* means that the dynamic choice of a parameter does not depend on the fitness landscape. The first to rigorously analyze a deterministic parameter control scheme are Jansen and Wegener [23]. They regard the performance of the (1+1) EA which uses in iteration t the mutation rate $2^k/n$, where $k \in \{1, 2, \ldots, 2^{\lceil \log_2 n \rceil - 2}\}$ is chosen such that $\log_2(k) \equiv t - 1 \pmod{\lceil \log_2 n \rceil - 1}$. In other words, they cyclically use the mutation rates $1/n, 2/n, \ldots, K/n$, where K is the largest power of two that is less than $n/2$. Jansen and Wegener demonstrate that there exists an example function where this dynamic EA significantly outperforms the (1+1) EA with any static mutation rate. However, they also observe that for many classic problems, this EA is slower by a factor of $\Theta(\log n)$.

In [32], a rank-based mutation rate was analyzed for the (μ+1) EA. A previous experimental study [5] suggested that this is a profitable approach, but the mathematical runtime analysis in [32] rather indicates the opposite. While there are artificial examples where a huge runtime gain could be shown and also the worst-case runtime of the (μ+1) EA reduces from essentially n^n to $O(3^n)$, a rigorous analysis on the OneMax function rather suggests that the high rate of offspring generated with a mutation rate much higher than $1/n$ brings a significant risk of slowing down the optimization process.

For two non-standard settings in evolutionary computation, deterministic parameter control mechanisms also gave interesting results. For problems where the solution length is not known [4], more precisely, where the number or the set of bits relevant for the solution quality is unknown, again random mutation rates gave good results [14, 19]. Here however, not a power-law scheme was used, but rather one based on very slowly decreasing summable sequences. For problems where the discrete variables take many values, e.g., the search space is $\{0, \dots, r-1\}^n$ for some large r, the question is how to change the value of an individual variable. The results in [15] suggest that a harmonic mutation strength, that is, changing a variable value by $\pm i$ with i chosen randomly with probability proportional to $1/i$, can be beneficial. This distribution was analyzed earlier in [7] for the one-dimensional case, where it was also shown to give the asymptotically best performance on a OneMax type problem.

For randomized search heuristics outside evolutionary computation, Wegener [33] showed that simulated annealing (using a time-dependent temperature) can beat the Metropolis algorithm (using a static temperature).

2.2.2 Adaptive Parameter Control

A parameter control scheme is called *adaptive* if it used some kind of feedback from the optimization process. This can be functionally dependent (e.g., the mutation rate depends on the fitness of the parent) or success-based (e.g., a 1/5th rule).

The first to conduct a runtime analysis for an adaptive parameter control mechanism (and show a small advantage over static choices) were Böttcher, Doerr, and Neumann [2]. They proposed to use the *fitness-dependent* mutation rate of $1/(\textsc{LeadingOnes}(x)+1)$ for the optimization of the LeadingOnes test function. They proved that with this choice, the runtime of the (1+1) EA improves to roughly $0.68n^2$ compared to a time of $0.86n^2$ stemming from the classic mutation rate $1/n$ or a runtime of $0.77n^2$ stemming from the asymptotically optimal static rate of approximately $1.59/n$.

For the $(1+(\lambda,\lambda))$ GA, a fitness-dependent offspring population size of order $\lambda = \Theta(\sqrt{n/d(x)})$ was suggested in [13], where $d(x)$ is the fitness-distance of the parent individual to the optimum. This choice improves the optimization time (number of fitness evaluations until the optimum is found) on OneMax from $\Theta(n\sqrt{\log(n)\log\log\log(n)/\log\log(n)})$ stemming from the optimal static parameter choice [9] to $O(n)$. Since in this adaptive algorithm the mutation rate p is functionally dependent on the offspring population size, namely via $p = \lambda/n$, the dynamic choice of λ is equivalent to a fitness-dependent mutation rate of $1/\sqrt{nd(x)}$.

In the aforementioned work by Badkobeh et al. [1], a fitness-dependent mutation rate of $\max\{\frac{\ln \lambda}{n \ln(en/d(x))}, \frac{1}{n}\}$ was shown to improve the classic runtime of $O(\frac{n \log\log \lambda}{\log \lambda} + \frac{n \log n}{\lambda})$ to $O(\frac{n}{\log \lambda} + \frac{n \log n}{\lambda})$. In [17], the (1+1) EA using a k-bit flip mutation operator together with a fitness-dependent choice of k was shown to give a performance on OneMax that is very close to the theoretical optimum (among all unary unbiased black-box algorithms), however, this differs only by lower order terms from the performance of the simple randomized local search heuristic (RLS). For nature-inspired algorithms other than evolutionary ones, Zarges [34, 35] proved that fitness-dependent mutation rates can be beneficial in artificial immune systems.

2.3 Self-adjusting and Self-adaptive Parameter Control

While all these results show an advantage of an adaptive parameter setting, it remains questionable if an algorithm user would be able to find such a functional dependence of the parameter on the fitness. This difficulty can be overcome via *self-adjusting* parameter choices, where the parameter is modified according to a simple rule often based on the success or progress of previous iterations, or via *self-adaptation*, where the parameter is encoded in the genome and thus subject to variation and selection. The understanding of self-adaptation is still very limited. The only theoretical work on this topic [6], however, is promising and shows examples where self-adaptation can lead to significant speed-ups for non-elitist evolutionary algorithms.

In contrast to this, the last years have produced a profound understanding of self-adjusting parameter choices. The first to perform a mathematical analysis were Lässig and Sudholt [28], who considered the (1+λ) EA and a simple parallel island model together with two self-adjusting mechanisms for population size or island number, including halving or doubling it depending on whether the current iteration led to an improvement or not. These mechanisms were proven to give significant improvements of the "parallel" runtime (number of generations) on various test functions without increasing significantly the "sequential" runtime (number of fitness evaluations).

In [10] it was shown that the fitness-dependent choice of λ for the $(1 + (\lambda, \lambda))$ GA described above can also be found in a self-adjusting way. To this aim, another success-based mechanism was proposed, which imitates the 1/5-th rule from evolution strategies. With some modifications, this mechanism also works on random satisfiability problems [3]. For the problem of optimizing an r-valued OneMax function, a self-adjustment of the step size inspired by the 1/5-th rule was found to find the asymptotically best possible runtime in [16].

These results indicate that success-based dynamics work well for adjusting parameters when a monotonic relation like "if progress is difficult, then increase the population size" holds. For adjusting a parameter like the mutation rate, it is less obvious how to do this. For example, in the search space $\{0, 1\}^n$ both a too large mutation rate (creating a stronger drift towards a Hamming distance of $n/2$ from the optimum) and a too small mutation rate (giving a too small radius of exploration) can be detrimental. For this reason, to obtain a self-adjusting version of their result on the optimal number k to optimize OneMax via k-bit flips [17], in [18] a learning mechanism was proposed that from the medium-term past estimates the efficiency of different parameter values. As shown there, this does find the optimal mutation strength sufficiently well to obtain essentially the runtime stemming from the fitness-dependent mutation strength exhibited before.

In the light of these works, our result from the methodological perspective shows that some of the difficulties of the learning mechanism of [18], e.g., the whole bookkeeping being part of it and also the setting of the parameters regulating how to discount information over time, can be overcome by the mechanism proposed in this work. In a sense, the use of larger populations enables us to adjust the mutation rate solely on information learned in the current iteration. However, we do also use the idea of [18] to intentionally use parameter settings which appear to be slightly off the current optimum to gain additional insight.

3 Preliminaries

3.1 Algorithm

We consider the (1+λ) EA with self-adjusting mutation rate for the minimization of pseudo-boolean functions $f : \{0,1\}^n \to \mathbb{R}$, defined as Algorithm 1.

The general idea of the mutation scheme is to adjust the mutation strength according to its success in the population. We perform mutation by applying standard bit mutation with two different mutation probabilities $r/(2n)$ and $2r/n$ and we call r the *mutation rate*. More precisely, for an even number $\lambda \geq 2$ the algorithm creates $\lambda/2$ offspring with mutation rate $r/2$ and with $2r$ each.

The mutation rate is adjusted after each selection. With probability a half, the new rate is taken as the mutation rate that the best individual (i. e. the one with the lowest fitness, ties broken uniformly at random) was created with (*success-based adjustment*). With the other 50% probability, the mutation rate is adjusted to a random value in $\{r/2, 2r\}$ (*random adjustment*). Note that the mutation rate is adjusted in each iteration, that is, also when all offspring are worse than the parent and thus the parent is kept for the next iteration.

If an adjustment of the rate results in a new rate r outside the interval $[2, n/4]$, we replace this rate with the corresponding boundary value. Note that in the case of $r < 2$, a subpopulation with rate less than 1 would be generated, which means flipping less than one bit in expectation. At a rate $r > n/4$, a subpopulation with rate larger than $n/2$ would be created, which again is not a very useful choice.

We formulate the algorithm to start with an initial mutation rate r^{init}. The only assumption on r^{init} is to be greater than or equal to 2. The (1+λ) EA with this self-adjusting choice of the mutation rate is given as pseudocode in Algorithm 1.

Algorithm 1 (1+λ) EA with two-rate standard bit mutation

```
Select $x$ uniformly at random from $\{0,1\}^n$ and set $r \leftarrow r^{\text{init}}$.
for $t \leftarrow 1, 2, \dots$ do
    for $i \leftarrow 1, \dots, \lambda$ do
        Create $x_i$ by flipping each bit in a copy of $x$ independently with probability $r_t/(2n)$ if $i \le \lambda/2$ and with probability $2r_t/n$ otherwise.
    $x^* \leftarrow \arg\min_{x_i} f(x_i)$ (breaking ties randomly).
    if $f(x^*) \le f(x)$ then
        $x \leftarrow x^*$.
    Perform one of the following two actions with prob. 1/2:
    • Replace $r_t$ with the mutation rate that $x^*$ has been created with.
    • Replace $r_t$ with either $r_t/2$ or $2r_t$, each with probability 1/2.
    Replace $r_t$ with $\min\{\max\{2, r_t\}, n/4\}$.
```

Let us explain the motivation for the random adjustments of the rate. Without such random adjustments, the rate can only be changed into some direction if a winning offspring is generated with this rate. For simple functions like OneMax, this is most likely sufficient. However, when the fitness of the best of $\lambda/2$ offspring, viewed as a function of the rate, is not unimodal, then several adjustments into a direction at first not yielding good offspring might be needed to reach good values of the rate. Here, our random adjustments enable the algorithm to cross such a valley of unfavorable rate values. We note that such ideas are not uncommon in evolutionary computation, with standard-bit mutation being the most prominent example (allowing to perform several local-search steps in one iteration to cross fitness valleys).

A different way to implement a mechanism allowing larger changes of the rate to cross unfavorable regions would have been to not only generate offspring with rates $r/2$ and $2r$, but to allow larger deviations from the current rate with some small probability. One idea could be choosing for each offspring independently the rate $r2^{-i}$ with probability $2^{-|i|-1}$ for all $i \in \mathbb{Z}$, $i \neq 0$. This should give similar results, but to us the process appears more chaotic (e.g., with not the same number of individuals produced with rates $r/2$ and $2r$).

The *runtime*, also called the *optimization time*, of the (1+λ) EA is the smallest t such that an individual of minimum f-value has been found. Note that t corresponds to a number of iterations (also called generations), where each generation creates λ offspring. Since each of these offspring has to be evaluated, the number of function evaluations, which is a classical cost measure, is by a factor of λ larger than the runtime as defined here. However, assuming a massively parallel architecture that allows for parallel evaluation of the offspring, counting the number of generations seems also a valid cost measure. In particular, a speed-up on the function $\textsc{OneMax}(x_1, \dots, x_n) := x_1 + \dots + x_n$ by increasing λ can only be observed in terms of the number of generations. Note that for reasons of symmetry, it makes no difference whether OneMax is minimized (as in the present paper) or maximized (as in several previous research papers).

Throughout the paper, all asymptotic notation will be with respect to the problem size n.

3.2 Drift Theorems

Our results are obtained by drift analysis, which is also used in previous analyses of the (1+λ) EA without self-adaptation on OneMax and other linear functions [11, 22].

The first theorems stating upper bounds on the hitting time using variable drift go back to [25, 30]. We take a formulation from [29] but simplify it to Markov processes for notational convenience.

Theorem 2 (Variable Drift, Upper Bound). *Let $(X_t)_{t\geq 0}$, be random variables describing a Markov process over a finite state space $S \subseteq \{0\} \cup [x_{\min}, x_{\max}]$, where $x_{\min} > 0$. Let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. If there exists a monotone increasing function $h(x)\colon [x_{\min}, x_{\max}] \to \mathbb{R}^+$, where $1/h(x)$ is integrable on $[x_{\min}, x_{\max}]$, such that for all $x \in S$ with $\Pr(X_t = x) > 0$ we have*

$$E(X_t - X_{t+1} \mid X_t = x) \geq h(x)$$

then for all $x' \in S$ with $\Pr(X_0 = x') > 0$

$$E(T \mid X_0 = x') \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{x'} \frac{1}{h(x)}\,\mathrm{d}x.$$

The variable drift theorem is often applied in the special case of *additive drift* in discrete spaces: assuming $E(X_t - X_{t+1} \mid X_t = x; X_t > 0) \geq \epsilon$ for some constant ϵ, one obtains $E(T \mid X_0 = x') \leq x'/\epsilon$.

Since we will make frequent use of it in the following sections as well, we will also give the version of the *Multiplicative Drift Theorem* for upper bounds, due to [12]. Again, this is implied by the previous variable drift theorem.

Theorem 3 (Multiplicative Drift [12]). *Let $(X_t)_{t\geq 0}$ be random variables describing a Markov process over a finite state space $S \subseteq \mathbb{R}_0^+$ and let $x_{\min} := \min\{x \in S \mid x > 0\}$. Let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. If there exist $\delta > 0$ such that for all $x \in S$ with $\Pr(X_t = x) > 0$ we have*

$$E(X_t - X_{t+1} \mid X_t = x) \geq \delta x \ ,$$

then for all $x' \in S$ with $\Pr(X_0 = x') > 0$,

$$E(T \mid X_0 = x') \leq \frac{1 + \ln\left(\frac{x'}{x_{\min}}\right)}{\delta} \ .$$

3.3 Chernoff Bounds

For reasons of self-containedness and as a courtesy to the reader, we state two well-known multiplicative Chernoff bounds and a lesser known additive Chernoff bound that is also known in the literature as Bennett's inequality.

Theorem 4 (Bennett's inequality, Chernoff Bounds [8, Theorem 1.12, Theorem 1.10])**.** *Let $X_1, \dots, X_n$ be independent random variables and let $X = \sum_{i=1}^{n} X_i$. Furthermore, let b such that $X_i \leq E(X_i) + b$ for all $i = 1, \dots, n$ and $\sigma^2 = \frac{1}{n}\sum_{i=1}^{n} \operatorname{Var}(X_i)$. Then, for all $\gamma > 0$*

$$\Pr(X \geq E(X) + \gamma) \leq \left(-\frac{\gamma}{b}\left(\left(1+\frac{n\sigma^2}{b\gamma}\right)\ln\left(1+\frac{b\gamma}{n\sigma^2}\right)-1\right)\right) .$$

Moreover, if the X_i for all $i = 1, \dots, n$ take values in $[0, 1]$ then

- $\Pr(X \leq (1-\delta)E(X)) \leq \exp(-\delta^2 E(X)/2)$ *for all* $0 \leq \delta \leq 1$.
- $\Pr(X \geq (1+\delta)E(X)) \leq \exp(-\delta^2 E(X)/(2+\delta))$ *for all* $\delta > 0$.

3.4 Occupation Probabilities

As mentioned above, we will be analyzing two depending stochastic processes: the random decrease of fitness and the random change of the mutation rate. Often, we will prove by drift analysis that the rate is drifting towards values that yield an almost-optimal fitness decrease. However, once the rate has drifted towards such values, we would also like the rates to stay in the vicinity of these values in subsequent steps. To this end, we apply the following theorem from [27]. Note that in the paper a slightly more general version including a self-loop probability is stated, which we do not need here.

Theorem 5 (Theorem 7 in [27])**.** *Let a Markov process $(X_t)_{t \geq 0}$ on $\mathbb{R}_0^+$, where $|X_t - X_{t+1}| \leq c$, with additive drift of at least d towards 0 be given (i. e., $E(X_t - X_{t+1} \mid X_t; X_t > 0) \geq d$), starting at 0 (i.e. $X_0 = 0$). Then we have, for all $t \in \mathbb{N}$ and $b \in \mathbb{R}_0^+$,*

$$\Pr(X_t \geq b) \leq 2e^{\frac{2d}{3c}(1-b/c)}.$$

We can readily apply this theorem in the following lemma that will be used throughout the paper to bound the rate r_t.

Lemma 2. *If there is a point $c \geq 4$ such that $\Pr(r_{t+1} < r_t \mid r_t > c) \geq 1/2 + \epsilon$ for some constant $\epsilon > 0$, then for all $t' \geq \min\{t \mid r_t \leq c\}$ and all $b \geq 4$ it holds $\Pr(r_{t'} \geq c + 2^b) \leq 2e^{-2b\epsilon/3}$.*

Proof. Apply Theorem 5 on the process $X_t := \max\{0, \lceil \log_2(r_t/c) \rceil\}$. Note that this process is on $\mathbb{N}_0$, moves by an absolute value of at most 1 and has drift $E(X_t - X_{t+1} \mid X_t; X_t > 0) = 2\epsilon$. We use $c := 1$ and $d := 2\epsilon$ in the theorem and estimate $1 - b \leq -b/2$. □

4 Far Region

In this first of three technical sections, we analyze the optimization behavior of our self-adjusting (1+λ) EA in the regime where the fitness distance k is at least $n/\ln\lambda$. Since we are relatively far from the optimum, it is relatively easy to make progress. On the other hand, this regime spans the largest number of fitness levels (namely $\Theta(n)$), so we need to exhibit a sufficient progress in each iteration. Also, this is the regime where the optimal mutation rate varies most drastically. Without proof, we remark that the optimal rate is n for $k \geq n/2 + \omega(\sqrt{n \log \lambda})$, $n/2$ for $k = n/2 \pm o(\sqrt{n \log \lambda})$, and then quickly drops to $r = \Theta(\log \lambda)$ for $k \leq n/2 - \varepsilon n$. Despite these difficulties, our (1+λ) EA manages to find sufficiently good mutation rates to be able to reach a fitness distance of $k = n/\ln\lambda$ in an expected number of $O(n/\log\lambda)$ iterations.

Lemma 3. *Let n be sufficiently large and $0 < k < n/2$. We define $c_1(k) = (2\ln(en/k))^{-1}$ and $c_2(k) = 4n^2/(n-2k)^2$.*

- *If $n/\ln\lambda \leq k$ and $r \leq c_1(k)\ln\lambda$, then the probability that a best offspring has been created with rate $2r$ is at least 0.5005.*
- *Let $\lambda \geq 50$. If $n/2 \geq r \geq c_2(k)\ln\lambda$, then the probability that all best offspring have been created with rate $r/2$ is at least 0.58.*
- *If $r \geq 2(1+\gamma)c_2(k)\ln\lambda$, then the probability that all best offspring are worse than the parent is at least $1-\lambda^{-\gamma}$.*

Proof. Let $Q(k,i,r)$ be the probability that standard bit mutation with mutation rate $p = r/n$ creates from a parent with fitness distance k an offspring with fitness distance at most $k-i$. Then

$$Q(k,i,r) = \sum_{x=i}^{k}\sum_{y=0}^{x-i}\binom{k}{x}\binom{n-k}{y}p^{x+y}(1-p)^{n-x-y}.$$

By comparing each component in $Q(k,i,r/2)$ and $Q(k,i,2r)$ we obtain

$$Q(k,i,2r)/Q(k,i,r/2) \geq 4^i\frac{(1-2r/n)^n}{(1-r/(2n))^n} \geq (1-o(1))4^i e^{-1.5r}.$$

Here we notice that $\ln(1-x) \geq -x - x^2$ for all $0 \leq x \leq 1/2$. Then

$$\left(\frac{1-\frac{2r}{n}}{1-\frac{r}{2n}}\right)^n = \left(1-\frac{1.5r}{n-0.5r}\right)^n \geq \exp\left(\frac{-1.5rn}{n-0.5r} - \frac{2(1.5r)^2n}{(n-0.5r)^2}\right) \geq (1-o(1))e^{-1.5r}.$$

The above inequality applies $\lambda = n^{O(1)}$. Therefore $r < \ln\lambda = O(\ln n)$. Since $Q(k,i,r)$ is monotone decreasing in i, let i^* be the largest i such that $Q(k,i,2r) \geq 4/\lambda$. We will then have $i^* \geq 2r$ because

$$Q(k,2r,2r) \geq \binom{k}{2r}(2p)^{2r}(1-2p)^n$$

$$\begin{aligned}&\geq \frac{2r\cdot 2r}{1\cdot 2}\cdot\frac{(k-1)(k-2)}{k^2}\cdot\left(\frac{k}{2r}\right)^{2r}\left(\frac{2r}{n}\right)^{2r}\cdot(1-o(1))e^{-2r}\\&> 2r\left(\frac{k}{en}\right)^{2r}\geq\left(\frac{k}{en}\right)^{2c_1(k)\ln\lambda}\geq\frac{4}{\lambda}.\end{aligned}$$

The second inequality which involves $(1-2p)^n$ again uses the fact that $r<\ln\lambda=O(\ln n)$. This means $p=o(1/\sqrt{n})$ and then we have $(1-2p)^n\geq\exp(-2pn-4p^2n)\geq(1-o(1))\exp(-2r)$. The $(1-o(1))$ factor and $(k-1)(k-2)/k^2$ is compensated by decreasing $(2r/2)$ to 1 if n is large enough. We notice that when $r=2$ we have $i^*\geq\ln(\lambda)/(2\ln\ln\lambda)$ since

$$Q(k,i,4/n)\geq\left(\frac{4k}{in}\right)^i e^{-4}\geq\left(\frac{1}{i\ln\lambda}\right)^i 4^i e^{-4}\geq\frac{4^i e^{-4}}{\lambda}\text{ for }i=\frac{\ln\lambda}{2\ln\ln\lambda}.$$

Let $\hat{\imath}=\ln(\lambda)/(2\ln\ln\lambda)$. We notice that

$$\frac{\partial p^{x+y}(1-p)^{n-x-y}}{\partial p}=p^{x+y-1}(1-p)^{n-x-y}\frac{np-x-y}{p-1}\geq 0\text{ when }x+y\geq i\geq np,$$

$Q(k,i,r)$ is increasing in r when $r\leq i$. We obtain $i^*\geq\hat{\imath}$ for all $r\geq 2$ which results in $4^{i^*}e^{-1.5r}\geq\exp(i^*(\ln 4-1.5/2))\geq\Theta(\lambda^{1/\ln\ln\lambda})$. We also need an upper bound on i^*. Since $r\leq\ln(\lambda)/2$ and $k\leq n/2$ implies $kr/n\leq\ln(\lambda)/4$, then Chernoff Bounds shows that the probability that $1.65\ln\lambda\geq 6.6kr/n$ bits being flipped from k bits is less than $\exp(-(5.6/7.6)1.4\ln\lambda)<1/\lambda$. This means $i^*<1.65\ln\lambda$. We use this to compute the upper bound on $Q(k,i^*,2r)$. Let $q(k,i,r)=Q(k,i,r)-Q(k,i+1,r)$ be the probability of that the fitness distance is decreased by i. We regard the terms in $q(k,i,r)$ where $x-y=i$. If x and y both increase by 1, the terms change by a factor of

$$\frac{k-x}{x+1}\cdot\frac{n-k-y}{y+1}\cdot\frac{p^2}{(1-p)^2}\leq(1+o(1))\frac{kp}{x+1}\cdot\frac{(n-k)p}{y+1}\leq\frac{r^2}{4xy}$$

If we consider $y\geq r$ then $xy>r^2$ and the factor $r^2/(4xy)\leq 1/4$. The sum of these factors for all $y\geq r$ is less than the geometric series with ratio $1/4$. Therefore, the sum from $0\leq y<r$ contributes to at least $2/3$ of the total sum $q(k,i,r)$. Consequently, if we look at the first $2r$ terms in $q(k,i^*,2r)$ and $q(k,i^*+1,2r)$ we have

$$\frac{q(k,i^*+1,2r)}{q(k,i^*,2r)}\geq\frac{2}{3}\cdot\frac{(k-2r-i^*+1)2p}{(2r+i^*)(1-2p)}\geq\frac{4kp}{3(2/2+1)i^*}=\frac{2rk}{3i^*n}.$$

Since $q(k,i^*+1,2r)\leq Q(k,i^*+1,2r)$ and $k\geq n/\ln\lambda$ we further more have

$$\frac{q(k,i^*,2r)}{Q(k,i^*+1,2r)}\leq\frac{3i^*\ln\lambda}{2r}\text{ and }\frac{Q(k,i^*,2r)}{Q(k,i^*+1,2r)}\leq 1+\frac{3i^*\ln\lambda}{2r}.$$

So finally $Q(k,i^*,2r)\leq 4(1+1.5i^*\ln\lambda/r)/\lambda$ and

$$Q(k,i^*,r/2)\leq\frac{Q(k,i^*,2r)}{4^{i^*}e^{-1.5r}}\leq\frac{4(1+\frac{1.5i^*\ln\lambda}{r})}{4^{i^*}e^{-1.5r}}\cdot\frac{1}{\lambda}.$$

If $\lambda = \omega(1)$ then the factor before $1/\lambda$ is $\Theta(\ln^2(\lambda)/\lambda^{1/\ln\ln(\lambda)}) = o(1)$. Otherwise if $\ln\lambda \geq 130$ or $r \geq 8$ or $r < 8$ but $i^* \geq 16$ we prove $Q(k, i^*, r/2) < 0.8/\lambda$ so that with probability less than 0.4 at least one offspring of $r/2$ achieve i^*. Then with probability at least $(1-(1-4/\lambda)^{\lambda/2}) \cdot (1-0.4) > 0.86 * 0.6 > 0.51$ all the best offspring have been created with $2r$. We first regard large λ. If $\ln\lambda \geq 130$, we define $t = i^*/r$ then $2 \leq t \leq (1.65\ln\lambda)/2$ and the factor $Q(k, i^*, r/2)/\lambda$ becomes

$$\frac{4(1+1.5t\ln\lambda)}{\exp(i^*(\ln 4 - 1.5/t))} \leq \frac{4(1+1.5t\ln\lambda)}{\exp(\hat{\imath}(\ln 4 - 1.5/t))}.$$

If $t \leq 3$ it is

$$\frac{4(1+1.5t\ln\lambda)}{\exp(\hat{\imath}(\ln 4 - 1.5/t))} \leq \frac{4(1+4.5\ln\lambda)}{\exp(\hat{\imath}(\ln 4 - 1.5/2))} < 0.8$$

otherwise it is

$$\frac{4(1+1.5t\ln\lambda)}{\exp(\hat{\imath}(\ln 4 - 1.5/t))} \leq \frac{4(1+1.5\ln^2\lambda)}{\exp(\hat{\imath}(\ln 4 - 1.5/3))} < 0.8.$$

If $\ln\lambda < 130$ and $r \geq 8$ we can bound

$$\frac{4(1+1.5i^*\ln(\lambda)/r)}{4^{i^*}e^{-1.5r}} \leq \frac{4(1+1.5\cdot 1.65\ln^2(\lambda)/r)}{\exp(2r\ln(4)-1.5r)} < \frac{4(1+41828/r)}{\exp(2r\ln(4)-1.5r)} < 0.8.$$

If $\ln\lambda < 130$, $r < 8$ but $i^* \geq 16$, then

$$\frac{4(1+1.5i^*\ln(\lambda)/r)}{4^{i^*}e^{-1.5r}} \leq \frac{4(1+1.5\cdot 130\cdot 16/2)}{4^{16}e^{-12}} < 0.3.$$

For $\ln\lambda < 130$ and $r < 8$, we prove that the probability that the best progress i attains $1.5r$ is at least $1 - 3\cdot 10^{-5}$. Conditioning on $1.5r \leq i \leq i^*$ it is obvious that i obtained from $2r$ with probability at least 0.5 since $q(k, i, 2r)/(q, i, r/2) = (1-o(1))4^i e^{-1.5r} > 1$. For the remaining $i \geq i^* + 1$ which has probability no less than $1.2\cdot 10^{-3}$, its very likely that a best offspring is from $2r$. This can cancel $3\cdot 10^{-5}$. We first compute $Q(k, 1.5r, 2r) > 21/\lambda$ in the following way. Since $1.5r < 12$ then for n large enough we have

$$Q(k, 1.5r, 2r) \geq (1-o(1))\binom{k}{1.5r}\left(\frac{2r}{n}\right)^{1.5r} e^{-2r} \geq (1-o(1))\frac{(2r)^{1.5r}}{(1.5r)!}\left(\frac{n}{k}\right)^{0.5r}\left(\frac{k}{en}\right)^{2r},$$

with $(k/(en))^{2r} \geq (k/(en))^{2c_1(k)\ln\lambda} = 1/\lambda$, $(2r)^{1.5r}/((1.5r)!) \geq 4^3/6$ and $(n/k)^{0.5r} \geq 2$. This means the best progress among $\lambda/2$ offspring attains $1.5r$ is at least $1 - (1-21/\lambda)^{\lambda/2} > 1 - 3\cdot 10^{-5}$. For $i \geq i^* + 1$ according to the definition of i^* and $Q(k, i^*, 2r)/Q(k, i^*+1, 2r) \leq 1 + 1.5i^*\ln\lambda/r$

$$4 > \frac{Q(k, i^*+1, 2r)}{\lambda} \geq \frac{4}{1+1.5i^*\ln\lambda/r} \geq \frac{4}{1+1.5\cdot 16\cdot 130/2} > \frac{1}{391}.$$

Then for $\lambda/2$ offspring we have $1-(1-Q(k,i^*+1,2r))^{\lambda/2} \geq 1.2\cdot 10^{-3}$. Moreover, use the fact that $i^*+1 \geq 2r+1$ we also have

$$\frac{Q(k,i^*+1,2r)}{Q(k,i^*+1,r/2)} \geq \frac{4^{2r+1}}{e^{1.5r}} > \frac{4^5}{e^3} > 50.$$

Therefore it is easy to bound

$$\frac{1-(1-Q(k,i^*+1,2r))^{\lambda/2}}{1-(1-Q(k,i^*+1,r/2))^{\lambda/2}} > \frac{1-(1-Q(k,i^*+1,2r))^{\lambda/2}}{Q(k,i^*+1,r/2)\cdot\lambda/2} > \frac{1-(1-Q(k,i^*+1,2r))^{\lambda/2}}{Q(k,i^*+1,2r)/50\cdot\lambda/2}.$$

We look at function $f_\alpha(x) := (1-(1-x)^\alpha)/(\alpha x)$ and $g_\alpha(x) := (1-x)^\alpha(1+\alpha x)$. Since $g'_\alpha(x) < 0$ for all $\alpha > 1$ and $0 < x < 1$, we have $g_\alpha(x) < g_\alpha(0) = 1$ when $0 < x < 1$. Therefore $f'_\alpha(x) = (g_{\alpha-1}(x)-1)/(\alpha x^2) < 0$ and

$$\frac{1-(1-Q(k,i^*+1,2r))^{\lambda/2}}{Q(k,i^*+1,2r)/50\cdot\lambda/2} > \frac{f_{\lambda/2}(4/\lambda)}{50} = \frac{1-(1-4/\lambda)^{\lambda/2}}{(4/\lambda)/50\cdot\lambda/2} \geq \frac{1-1/e^2}{1/25} > 21.$$

This means that if the best offspring made a progress of i^*+1 or more, then the conditional probability that it is from the $2r$-subpopulation is at least $21/22$. Finally we can bound the probability that a best offspring is from the $2r$-subpopulation by

$$(1-3\cdot 10^{-5}-1.2\cdot 10^{-3})\cdot 0.5 + 1.2\cdot 10^{-3}\cdot\frac{21}{22} > 0.5005.$$

For the second statement, let $X(k,r)$ denote the random decrease of the fitness distance when apply standard bit mutation with probability $p = r/n$ to an individual with k ones. Then $E(X) = kp-(n-k)p = (2k-n)p$. According to Bennett's inequality (Theorem 4), for any $\Delta > 0$ we have

$$\Pr(X \geq E(X)+\Delta) \leq \exp\left(-\mathrm{Var}(X)\cdot h\left(\frac{\Delta}{\mathrm{Var}(X)}\right)\right)$$

where $h(u) = (1+u)\ln(1+u)-u$. We compare $h(u)$ with τu^2 for any constant factor τ. Let $g(u) = h(u)-\tau u^2$ be the difference, then $g(0) = g'(0) = 0$ while $g''(u) = 1/(1+u)-2\tau \geq 0$ when $1/(1+u) \geq 2\tau$. This means $h(u) \geq u^2/(2u+2)$. We now apply this bound with $X = X(k,2r)$ and $\Delta = E(X(k,r/2)) - E(X(k,2r)) = (n-2k)1.5r/n > 0$. We have $\mathrm{Var}(X(k,2r)) = n(2p)(1-2p) = 2r(1-2r/n)$ and $\Delta/\mathrm{Var} = (3/4)(n-2k)/(n-2r)$. Then,

$$\begin{aligned}\Pr(X(k,2r) \geq E(X(k,r/2))) &\leq \exp\left(-\frac{\Delta^2}{(2+2u)\mathrm{Var}(X(k,2r))}\right)\\ = \exp\left(-\frac{9(n-2k)^2r}{4n(7n-8r-6k)}\right) &\leq \exp\left(-\frac{9(n-2k)^2c_2(k)\ln\lambda}{28n^2}\right) = \frac{1}{\lambda^{9/7}}.\end{aligned}$$

We notice that we have $7n-8r-6k > 7n-4n-3n = 0$ in the second inequality. Therefore, with probability less than $\lambda^{-9/7}(\lambda/2) < 50^{-2/7}/2 < 0.2$ the best offspring of rate $2r$ is better than the expectation of rate $r/2$. For rate $r/2$, let X^+ and X^- be the number of one-bits flipped and zero-bits flipped, respectively. Both X^+ and X^- follow a binomial distribution and $X = X^+ - X^-$. We know $E(X^+) = kr/(2n)$ and $E(X^-) = (n-k)r/(2n) \geq \ln\lambda$. The median of X^+ is between $\lfloor E(X^+)\rfloor$ to $\lceil E(X^+)\rceil$ by [26]. This means $\Pr(X^+ \geq E(X^+) - 1) \geq \Pr(X^+ \geq \lfloor E(X^+)\rfloor) \geq 1/2$. For $\ln\lambda$ large enough, we can use a normal distribution to approximate X^- and have $\Pr(X^- \leq E(X^-) - 1) = 1/2 - o(1)$. Otherwise we notice that $\ln\lambda \geq \ln 50 > 3$ and $r/(2n) \leq 1/4$. Let real number t denote $E(X^-)$ and m denote $n-k$. It is clear that

$$\begin{aligned}\Pr(X^- \leq t-1) &= \Pr(X^- \leq \lfloor t-1\rfloor) \geq \Pr(X^- \leq \lceil t\rceil - 2)\\ &= \Pr(X^- \leq \lceil t\rceil) - \Pr(X^- = \lceil t\rceil) - \Pr(X^- = \lceil t\rceil - 1).\end{aligned}$$

We see that $\Pr(X^- \leq \lceil t\rceil) \geq 1/2$ and

$$\frac{\Pr(X^- = \lceil t\rceil)}{\Pr(X^- = \lceil t\rceil - 1)} = \frac{(m-t+1)t}{\lceil t\rceil \cdot m \cdot (1 - r/(2n))} < \frac{t}{\lceil t\rceil (1 - r/(2n))} < \frac{1}{1 - r/(2n)} < \frac{4}{3}$$

and

$$\frac{\Pr(X^- = \lceil t\rceil - 1)}{\Pr(X^- = \lceil t\rceil - 2)} = \frac{(m-t+2)t}{(\lceil t\rceil - 1) \cdot m \cdot (1 - r/(2n))} < \frac{t}{(\lceil t\rceil - 1)(1 - r/(2n))} < \frac{16}{9}.$$

Then we obtain

$$\Pr(X^- \leq E(X^-) - 1) > \frac{1}{2} \cdot \frac{1}{1 + 16/9 + (16/9)(4/3)} > 0.097$$

Therefore $\Pr(X(k, r/2) \geq E(X(k, r/2))) \geq 0.048$, with probability at least $1 - (1 - 0.048)^{\lambda/2} > 0.7$ one offspring beats $E(X(k, r/2))$. Therefore all best offspring are from $r/2$ with probability at least $0.7 \cdot (1 - 0.2) > 0.56$, this proves the second statement of the lemma.

An offspring of mutation rate r/n is not worse than the parent if and only if $X(k, r) \geq 0$ and, again by using Bennett's inequality, we have

$$\begin{aligned}\Pr(X \geq 0) &\leq \exp\left(-\mathrm{Var}(X) \cdot h\left(\frac{-E(X)}{\mathrm{Var}(X)}\right)\right)\\ &\leq \exp\left(-\frac{(n-2k)^2 r}{2n(2n-2k-r)}\right) \leq \exp\left(-\frac{(n-2k)^2 r}{4n^2}\right).\end{aligned}$$

Therefore if $r = 2(1+\gamma)c_2(k)\ln\lambda$, then the probability above for $r/2$ is $1/\lambda^{1+\gamma}$ and for 2α is $1/\lambda^{4+4\gamma}$. This proves the third statement. □

The lemma above shows that the rate r is attracted to the interval $[c_1(k)\ln n, c_2(k)\ln n]$. Unfortunately, we cannot show that we obtain a sufficient progress in the fitness for exactly this range of r-values. However, we can do so for a range smaller only by constant factors. This is what we do now (for large values of k) and in Lemma 5 (for smaller values of k). This case distinction is motivated by the fact that $c_2(k)$ becomes very large when k approaches $n/2$. Having a good drift only for such a smaller range of r-values is not a problem since the random movements of r let us enter the smaller range with constant probability, see Theorem 6 and its proof.

Let $\Delta := \Delta(\lambda/2, k, r)$ denote the fitness gain after selection among the best of $\lambda/2$ offspring generated with rate r from a parent with fitness distance $k := \textsc{OneMax}(x)$. Let $x^{(i)}, i \in \{1, \dots, \lambda/2\}$, be independent offspring generated from x by flipping each bit independently with probability r/n. Then the random variable Δ is defined by $\Delta := \max\{0, k - \min\{\textsc{OneMax}(x^{(i)}) \mid i \in \{1, \dots, \lambda/2\}\}\}$.

We next show that a narrow region contained in $c_1(k) \ln \lambda$ and $c_2(k) \ln \lambda$ provides at least logarithmic drift on fitness. We first do the proof for large k because $c_2(k)$ will be $\omega(1)$ when k is close to $n/2$.

Lemma 4. *Let $2n/5 \leq k < n/2$ and n be large enough. Let c be such that $c \leq \min\{n^2/(50(n-2k)^2), n/(4\ln\lambda)\}$ and $c \geq 1/2$. Let $r = c\ln\lambda \geq 1$ and $\lambda \geq 2$, then $E(\Delta) \geq 10^{-3} \ln \lambda$.*

Proof. Note that if $k \geq 2n/5$ then $n^2/(50(n-2k)^2) \geq 1/(50 \cdot 0.2^2) = 1/2$. We look at the number X of flips in k ones. This random variable follows a binomial distribution $\mathrm{Bin}(k, p)$ where $p = r/n$. Assume $u = kp \geq 1$, $B(x) = \binom{k}{x} p^x (1-p)^{k-x}$ and $F(x) = \sum_{i \geq x}^{k} B(i)$. If we use a normal distribution to approximate X, it's not hard to see that the probability of hitting the mean satisfies $F(u) \geq \Omega(1)$. More specifically, if $u < 1$ then $F(u) = 1 - B(0) = 1 - (1-p)^n \geq 1 - 1/e > 1/2$. Otherwise if $u \geq 1$ then the worst case for $F(u)$ is $u = 1 + o(1)$ and $F(u) \geq 1 - (1-1/n)^n - (1-1/n)^{n-1} > 1/4$. We now estimate $F(u+\delta)$. If $F(u+\delta) < F(u)/2$ then $F(u) - F(u+\delta) \geq F(u)/2 = 1/8$. When comparing $F(u+\delta) - F(u+2\delta)$ to $F(u) - F(u+\delta)$ we first notice that $B(x+1)/B(x) = \frac{k-x}{x+1} \cdot \frac{p}{1-p}$ which means $B(0) < B(1) < \cdots < B(u) > B(u+1) > \cdots > B(k)$. Comparing $B(u)$ to $B(u+2\delta)$ we see that

$$\begin{aligned}\frac{B(u+2\delta)}{B(u)} &= \frac{(k-u)\cdots(k-u-2\delta+1)}{(u+1)\cdots(u+2\delta)} \cdot \frac{p^{2\delta}}{(1-p)^{2\delta}} \\ &\geq \left(\frac{k-u-2\delta}{k(1-p)}\right)^{2\delta} \frac{u^{2\delta}}{(u+1)\cdots(u+2\delta)}\end{aligned}$$

We take $\delta \leq \sqrt{u}/10$ for $u = \Theta(k)$ and $\delta \leq 1.4u$ otherwise for $u = o(k)$ and then bound

$$\left(\frac{k-u-2\delta}{k(1-p)}\right)^{2\delta} = \left(1 - \frac{2\delta}{k-u}\right)^{2\delta} \geq 0.98.$$

Using Stirling's approximation for $(u+1)\cdots(u+2\delta)$ we have

$$(u+2\delta)! \leq \sqrt{2\pi(u+2\delta)} \left(\frac{u+2\delta}{e}\right)^{u+2\delta} \exp\left(\frac{1}{12(u+2\delta)}\right),$$

and

$$u! \geq \sqrt{2\pi u} \left(\frac{u}{e}\right)^u \exp\left(\frac{1}{12u+1}\right).$$

Then

$$\frac{u!}{(u+2\delta)!} \geq (1-o(1))\sqrt{\frac{u}{u+2\delta}}\frac{u^u e^{2\delta}}{(u+2\delta)^{u+2\delta}},$$

and therefore

$$\begin{aligned}\frac{B(u+2\delta)}{B(u)} &\geq 0.98\sqrt{\frac{u}{u+2\delta}}\frac{u^{u+2\delta}e^{2\delta}}{(u+2\delta)^{u+2\delta}}\\ &\geq 0.98\sqrt{\frac{u}{u+2.8u}}\left(1-\frac{2\delta}{u+2\delta}\right)^{u+2\delta}e^{2\delta}\\ &\geq \frac{1}{2}\exp\left(\ln\left(\frac{u}{u+2\delta}\right)(u+2\delta)+2\delta\right)\\ &\geq \frac{1}{2}\exp\left(-\left(\frac{2\delta}{u}\right)^2 u\right).\end{aligned}$$

For $u = \Theta(k)$ we take $\delta = \sqrt{u}/10$, otherwise let $\delta = 0.49u/\sqrt{c}$ satisfying $\delta \leq 1.4u$ for $c > 0.5$ which is required by assumption. Then $(2\delta/u)^2\, u \leq 0.98\ln\lambda$ and $B(u+2\delta)/B(u) \geq 1/(2\lambda^{0.98})$. Then $F(u+\delta) > \delta B(u+2\delta) > \delta B(u)/(2\lambda^{0.98})$. Notice that $\delta B(u) \geq F(u) - F(u+\delta)$ therefore $F(u+\delta) \geq 1/(16\lambda^{0.98})$. With probability at least $\gamma = 1-(1-1/(16\lambda^{0.98}))^{\lambda/2} > 1/32 > 0.03$ we have that one offspring flips at least $u+\delta$ ones which shows that $E(\Delta) \geq \gamma(u+\delta-(n-k)p) \geq \gamma(\delta-(n-2k)p)$. We compare δ to $(n-2k)p$. If $u = o(k)$ then $\delta = 0.49kp/\sqrt{c} \geq 0.19np/\sqrt{c}$. If $c \leq n^2/(50(n-2k)^2)$ we have $\delta/((n-2k)p) \geq 0.19/\sqrt{1/50} \geq 1.34$ and $\delta-(n-2k)p \geq 0.25\delta$. Notice that $\gamma \geq 0.03$ then $E(\Delta) \geq 0.03\cdot 0.25\delta \geq 0.03\cdot 0.25\cdot 0.19\sqrt{c}\ln\lambda \geq 10^{-3}\ln\lambda$. Otherwise, if $u = \Theta(k)$ then $r = \Theta(n)$ will require $n^2/(n-2k)^2 \geq r/\ln\lambda$. This means $n-2k = O(\sqrt{n/\log\lambda})$ and $(n-2k)r/n = O(\sqrt{n/\log\lambda})$ which is of lower order compared to $\delta = \sqrt{u}/10$. Hence, $E(\Delta) \geq \ln\lambda$. □

We now extend the lemma to the whole region of $n/\ln\lambda \leq k < n/2$. If $k < 2n/5$ the situation becomes easier because $4 \leq c_2(k) < 100$ and every r in the smaller range $[c_1(k)\ln\lambda, \ln(\lambda)/2]$ provides at least an expected logarithmic fitness increase. Together with the previous lemma, we obtain the following statement for the drift in the whole region $n/\ln\lambda \leq k < n/2$.

Lemma 5. *Let $n/\ln\lambda \leq k < n/2$ and n be large enough. Assume $\lambda \geq 2$. If $r \in [c_1(k)\ln\lambda, c_2(k)\ln(\lambda)/200]$ for $k \geq 2n/5$ and $r \in [c_1(k)\ln\lambda, \ln(\lambda)/2]$ for $k < 2n/5$ with $c_1(k), c_2(k)$ defined as in Lemma 3, then $E(\Delta) \geq 10^{-3}\ln(\lambda)/\ln(en/k)$.*

Proof. For $r \leq \ln(\lambda)/2$ we consider the probability $Q(k,i,r)$ of creating from a parent with distance k an offspring with fitness distance at least $k-i$ and $i := c_1(k)\ln\lambda = 0.5\ln(\lambda)/\ln(en/k)$, via standard bit mutation with probability r/n.

$$Q(k,i,r) \geq \binom{k}{i}(p)^i(1-p)^n \geq \left(\frac{k}{i}\cdot\frac{r}{n}\right)^i e^{-r} \geq \left(\frac{k}{en}\right)^i e^{-r} \geq e^{-\ln(\lambda)/2}e^{-r} \geq e^{-\ln\lambda} = \frac{1}{\lambda}.$$

In the second greater or equal symbol we apply $(1-p)^n \geq \exp(-pn-p^2n) \geq (1-o(1))\exp(-r)$ for p satisfying $p^2n = o(1)$. The $(1-o(1))$ factor will be compensated by using $(k/i)^i$ to estimate $\binom{k}{i}$. Hence, $\Pr(\Delta \geq c_1(k)\ln\lambda) \geq 1-(1-1/\lambda)^{\lambda/2} \geq 1-e^{-1/2} > 0.3$ and consequently $E(\Delta) > 0.15\ln(\lambda)/\ln(en/k)$. For $r > \ln(\lambda)/2$, note that this occurs only for $k > 2n/5$. In this case we apply Lemma 4 and obtain $E(\Delta) > 10^{-3}\ln\lambda \geq 10^{-3}\ln(\lambda)/\ln(en/k)$. □

If we only consider generations that use a rate within the right region, we can bound the expected runtime to reach $k \leq n/\ln\lambda$ by $O(n/\log\lambda)$ since the drift on the fitness is of order $\log\lambda$. The following theorem shows that the additional time spent with adjusting the rate towards the right region does not change this bound on the expected runtime.

Theorem 6. *The (1+λ) EA needs $O(n/\log\lambda)$ generations in expectation to reach a* OneMax*-value of $k \leq n/\ln\lambda$ after initialization.*

Proof. We first argue quickly that it takes an expected number of at most $O(\sqrt{n})$ iterations to reach a fitness distance of $k < n/2$. To this end, we note that if $k \geq n/2$, then the probability for an offspring to have a strictly better fitness than the parent is at least $\Theta(1)$ for any $2 \leq r \leq n/4$. Consequently, the expected fitness gain in one iteration is at least constant. The initial fitness distance k deviates from $n/2$ by $\Omega(\sqrt{n})$ in expectation. Hence, it takes $O(\sqrt{n})$ generations to obtain $k < n/2$.

Without loss of generality we assume $k < n/2$ for the initial state. Our intuition is that once we begin to use rate r bounded by $c_1(k)\ln\lambda$ and $c_2(k)\ln\lambda$ at some distance level k, we will have a considerable drift on the OneMax-value and the strong drift on the rate keeps r within or not far away from the bounds. After we make progress and k decreases to a new level, the corresponding c_1 and c_2 decrease, and the algorithm takes some time to readjust r into new bounds.

We consider the stochastic process $X_t = \log_2(r_t)$ and the current OneMax-value Y_t. According to Lemma 3 we have $E(X_t - X_{t+1} \mid X_t; X_t > \log_2(c_2(Y_t)\ln\lambda)) \geq \epsilon = \Omega(1)$ and $E(X_{t+1} - X_t \mid X_t; X_t < \log_2(c_1(Y_t)\ln\lambda)) \geq \epsilon = \Omega(1)$. We pessimistically assume that all the iterations adjusting r_t make no progress. Let $k_0 > k_1 > \cdots > k_N$ be the OneMax-values taken by Y_t until k_N hits $n/\ln\lambda$. According to the additive drift analysis from Theorem 2 it takes at most $O(\log n)$ iterations to bound $r_t/\ln\lambda$ by $c_1(k_0)$ and $c_2(k_0)$, no matter how we set the initial rate. Consider t_i to be the last time that $Y_t = k_i$. This means r_t makes a progress of $k_i - k_{i+1} > 0$. Referring to Lemma 3 we know that $r_t \leq 2(1+\gamma)c_2(k_i)\ln\lambda$ with probability at least $1-\lambda^{-\gamma}$. For $r_t \leq 2(1+\gamma)c_2(k_i)\ln\lambda$, according to the additive drift theorem, it takes an expected number of at most $O(\log(2(1+\gamma)c_2(k_i)/c_2(k_{i+1})))$ iterations to reach $r_{t+1} \leq c_2(Y_t)\ln\lambda$. When we sum up the different expectation on $\gamma = 1, 2, \ldots$, the increase of $\log(1+\gamma)$ can be neglect comparing to the probability decrease on $(1-\lambda^{-\gamma})'$. Thus we can bound this runtime by $O(\log(4c_2(k_i)/c_2(k_{i+1})))$. The total number of iterations of the readjustment process for r_t to satisfy the upper bound is

$$\sum_{i=0}^{N-1} O\left(\log\left(\frac{4c_2(k_i)}{c_2(k_{i+1})}\right)\right) = O\left(\log\left(4^N\frac{c_2(k_0)}{c_2(k_N)}\right)\right) = O(N + \log n).$$

We then consider the iterations of the readjustment process for r_t to satisfy the lower bound. Since $c_1(k_i)$ decreases along with k_i, once $c_1(k_0)$ is hit, the lower bound condition is obtained for all the following k_i.

Now we compute the expected number of generations for k_0 to decrease to k_N. We choose b large enough such that $2e^{-2b\epsilon/3} \leq 1/2 - \delta/2$ holds for some positive constant $\delta > 0$ and note that b is constant. Applying Lemma 2 we obtain $\Pr(r_t \geq c_2(k)\ln\lambda + 2^b) \leq 2e^{-2b\epsilon/3}$ and $\Pr(r_t \leq c_1(k)\ln\lambda - 2^b) \leq 2e^{-2b\epsilon/3}$. Once r_t is between $c_1(k)\ln\lambda$ and $c_2(k)\ln\lambda$, before k decreases to another bound level, we have that $c_1(k)\ln\lambda - 2^b \leq r_t \leq c_2(k)\ln\lambda + 2^b$ happens with probability at least δ. We see that there are at most $\log_2^{200}$ steps between range $c_1(k)\ln\lambda \leq r \leq c_2(k)\ln\lambda$ and an even smaller range $c_1(k)\ln\lambda \leq r \leq c_2(k)\ln(\lambda)/200$ which is described in Lemma 5. If r_t reaches the wider region, it takes at most a constant number of iterations α in expectation to reach the narrow region because our mutation scheme employs a 50% chance to perform a random step of the mutation rate. Based on Lemma 5 the narrow region for the rate ensures $0.05\ln(\lambda)/\ln(en/k)$ drift on the fitness at distance k. This contributes to an average drift of at least $0.05\ln(\lambda)/\ln(en/k)\cdot\delta/(1+\alpha) = \Omega(\log(\lambda)/\log(en/k)$ for all random rates at distance k. Applying Theorem 2, we can estimate the runtime as

$$O\left(\frac{1}{\log(\lambda)/\log(e\log\lambda)} + \int_{n/\log\lambda}^{n/2} \frac{\mathrm{d}k}{\log(\lambda)/\log(en/k)}\right) = O\left(\frac{n}{\log\lambda}\right).$$

Details about how to compute the above integral can be found in the proof of Theorem 4 of [1]. We notice that N is the number of different k values and N must be bounded by the above runtime. Combining the expected number of $O(N + \log n)$ iterations to adjust r_t and the expected number of $O(\sqrt{n})$ iterations to hit $k < n/2$, the total runtime is $O(n/\log\lambda)$ in expectation. □

5 Middle Region

In this section we estimate the expected number of generations until the number of one-bits has decreased from $k \leq n/\ln\lambda$ to $k \leq n/\lambda$. We first claim that the right region for r is $1 \leq r \leq \ln(\lambda)/2$. Hence, the (1+λ) EA is not very sensitive to the choice of r here. Intuitively, this is due to the fact that a total fitness improvement of only $O(n/\log\lambda)$ suffices to cross the middle region, whereas an improvement of $\Omega(n)$ is needed for the far region.

We estimate the drift of the fitness in Lemma 6 and apply that result afterwards to estimate the number of generations to cross the region.

Lemma 6. *Let $n/\lambda \leq k \leq n/\ln\lambda$, $\lambda \geq 26$ and $1 \leq r \leq \ln(\lambda)/2$. Then*

$$E(\Delta) \geq \min\left\{\frac{1}{8}, \frac{\sqrt{\lambda}k}{32n}\right\}.$$

Proof. The probability that no zero-bit flipped in a single mutation is $(1 - r/n)^{n-k} \geq e^{-r} \geq 1/\sqrt{\lambda}$. We regard the number Z of offspring that have no flipped zeros. The expectation $E(Z)$ is at least $1/\sqrt{\lambda}\cdot\lambda/2 = \sqrt{\lambda}/2$. Applying Chernoff bounds (Theorem 4), we observe that Z exceeds $\lambda_0 := \sqrt{\lambda}/4$ with probability at least $1-\exp(-\sqrt{\lambda}/16) > 1/4$ since $\lambda \geq 26$. Assuming this to happen, we look at the first λ_0 offspring without flipped zeros. For $i \in \{1,\dots,\lambda_0\}$ let X_i be the number of flipped ones in the i-th offspring. Then X_i are i.i.d. with $X_i \sim \mathrm{Bin}(k, r/n)$. Let $X^* = \max\{X_i\}$. By applying a result on order statistics for binomially distributed random variables by Gießen and Witt [22, Lemma 4] we obtain the following:

$$\text{If } \lambda_0 kr/n \geq \alpha \text{ then } E(X^*) \geq \alpha/(1+\alpha).$$

Therefore, $\lambda_0 kr/n \geq 1$ implies $E(X^*) \geq 1/2$, otherwise $1+\alpha < 2$ implies $E(X^*) \geq \lambda_0 kr/(2n)$. Thus,

$$E(X^*) \geq \min\{1/2, \sqrt{\lambda}k/(8n)\}.$$

Hence, using the law of total probability, we obtain the lower bound on the drift for the middle region. □

We now use our result on the drift to estimate the time spent in this region. We notice that $c_2(k) = 4 + o(1)$ when $k = o(n)$. This means we will have frequently often $r_t \in [1, \ln(\lambda)/2]$ which provides the drift we need.

Theorem 7. *Let $\lambda \geq 26$. Assume $k \leq n/\ln\lambda$ for the current* OneMax*-value of the self-adjusting (1+λ) EA. Then the expected number of generations until $k \leq n/\lambda$ is $O(n/\log\lambda)$.*

Proof. For $k \leq n/\ln\lambda$ the upper bound from Lemma 3 is $c_2(k) = 4/(1-2/\ln\lambda) < 11$. According to the lemma we have for $X_t := \lceil \log_2 r_t \rceil$ that $E(X_t - X_{t+1} \mid X_t; X_t > \log_2(c_2(k)\ln\lambda)) \geq \epsilon = \Omega(1)$. The additive drift theorem yields that in $O(\log n)$ time we have $r_t \leq c_2(k)\ln\lambda$. We choose b large enough such that $2e^{-2b\epsilon/3} \leq 1-\delta$ holds for some positive constant $\delta > 0$ and note that b is constant. Applying Lemma 2 we obtain $\Pr(r_t \geq c_2(k)\ln\lambda + 2^b) \leq 2e^{-2b\epsilon/3}$ and $r_t \leq c_2(k)\ln\lambda + 2^b$ happens with probability at least δ. Once $r_t < c_2(k)\ln\lambda + 2^b < 11\ln\lambda + 2^b$ it takes at most a constant number of iterations α in expectation to draw r_t to $\ln(\lambda)/2$ or less. According to Lemma 6 this ensures a drift of $(1/4)\min\{1/2, \sqrt{\lambda}k/(8n)\}$, which implies an average drift of at least $c\min\{1, \sqrt{\lambda}k/n\}$ over all random rates at distance k, where $c > 0$ is a constant. The minimum is taken on the first argument if $k > n/\sqrt{\lambda}$, and on the second if $k < n/\sqrt{\lambda}$.

We are interested in the expected time to reduce the OneMax-value to a most n/λ. To ease the application of drift analysis, we artificially modify the process and make it create the optimum when the state (OneMax-value) is strictly less than n/λ. Clearly, the first hitting time of state at most n/λ does not change by this modification. Applying the variable drift theorem (Theorem 2) with $x_{\min} = n/\lambda$, $X_0 = k \leq n/\ln\lambda$ and $h(x) = c\min\{1, \sqrt{\lambda}k/n\}$, the expected number of generations to reach state at most n/λ is bounded from above by

$$\frac{n/\lambda}{c\sqrt{\lambda}(n/\lambda)/n} + \int_{n/\lambda}^{n/\sqrt{\lambda}} \frac{n}{c\sqrt{\lambda}x}\,\mathrm{d}x + \int_{n/\sqrt{\lambda}}^{n/\ln\lambda} \frac{1}{c}\,\mathrm{d}x = O\Big(\frac{n}{\sqrt{\lambda}}\Big) + O\Big(\frac{n\log\lambda}{\sqrt{\lambda}}\Big) + O\Big(\frac{n}{\log\lambda}\Big),$$

which is $O(n/\log\lambda)$. The overall expected number of generations spent is $O(\log n + n/\log\lambda) = O(n/\log\lambda)$ since $\lambda = n^{O(1)}$ by assumption. □

6 Near region

In the near region, we have $k \le n/\lambda$. Hence, the fitness is so low that we can expect only a constant number of offspring to flip at least one of the remaining one-bits. This assumes constant rate. However, higher rates are detrimental since they are more likely to destroy the zero-bits of the few individuals flipping one-bits. Hence, we expect the rate to drift towards constant values, as shown in the following lemma.

Lemma 7. *Let $k \le n/\lambda$, $\lambda \ge 45$ and $4 \le r_t \le \ln(\lambda)/4$. Then the probability that $r_{t+1} = r_t/2$ is at least 0.5099.*

Proof. To prove the claim we exploit the fact that only few one-bits are flipped in both subpopulations. Using $r := r_t$, we shall argue as follows. With sufficiently high (constant) probability, (i) the $2r$-subpopulation contains no individual strictly better than the parent, that is, with fitness less than k, and (ii) all $2r$-offspring with fitness r are identical to the parent. Conditional on this, either the $r/2$-population contains individuals with fitness less than k and the winning individual surely stems from this subpopulation, or the $r/2$-population contains no better offspring. In the latter case, we argue that there are many more individuals with fitness exactly k in the $r/2$-population than in the $2r$-population, which gives a sufficiently high probability for taking the winning individual from this side (as it is chose uniformly at random from all offspring with fitness k).

Let $N_{r/2}$ and N_{2r} be the number of offspring that did not flip any zero-bits using rate $r/2$ and $2r$, respectively. Then $E(N_{r/2}) = (\lambda/2)(1-r/(2n))^{n-k} \ge (1-o(1))\lambda e^{-r/2}/2$, since $k \ge 1$ and

$$\left(1-\frac{r}{2n}\right)^{n-1} \ge e^{-\frac{r}{2}}\left(1-\frac{r}{2n}\right)^{\frac{r}{2}-1} \ge e^{-\frac{r}{2}}\left(1-\frac{c\ln n}{8n}\right)^{\frac{c\ln n}{4}-1} = (1-o(1))e^{-\frac{r}{2}},$$

where we used that $r \le \ln\lambda/4$ and $\lambda = n^{O(1)}$, i.e. $\ln\lambda \le c\ln n$ for some constant c. Using $k \le n/\lambda$ we get $E(N_{2r}) = (\lambda/2)(1-2r/n)^{n-k} \le (\lambda/2)e^{-2r(1-1/\lambda)}$. In fact, we can discriminate N_1 and N_2 by using Theorem 4 in the following way: we have

$$\begin{aligned}\Pr\left(N_{r/2} \le \frac{\lambda}{4}e^{-\frac{r}{2}}\right) &\le \exp\left(-(1-o(1))^3\frac{\lambda}{16}e^{-\frac{r}{2}}\right)\\ &\le \exp\left(-(1-o(1))\frac{1}{16}\lambda^{7/8}\right)\\ &< 0.175,\end{aligned}$$

for sufficiently large n, since $r \le (\ln\lambda)/4$ and $\lambda \ge 45$. Similarly, we obtain

$$\Pr\left(N_{2r} \ge \lambda e^{-2r(1-\frac{1}{\lambda})}\right) \le \exp\left(-\frac{1}{6}\lambda^{1-\frac{1}{2}(1-\frac{1}{\lambda})}\right) < 0.312.$$

Note that $e^{-2r(1-1/\lambda)} < e^{-r/2}/4$ holds for all $r \geq 4$ and $\lambda \geq 45$. Since the offspring are generated independently, the events $N_{r/2} > \lambda e^{-r/2}/4$ and $N_{2r} < \lambda e^{-2r(1-\frac{1}{\lambda})}$ happen together with probability at least $(1-0.175)\cdot(1-0.312) \geq 0.567 =: 1-p_{\text{err}_1}$. Conditioning on this and by using a union bound the probability p_{err_2} that at least one of the N_{2r} offspring that do not flip any zero-bits flips at least one one-bit can be upper bounded by

$$p_{\text{err}_2} := N_{2r} \cdot \frac{2kr}{n} \leq 2re^{-2r\left(1-\frac{1}{\lambda}\right)} \leq 0.004$$

using $k/n \leq 1/\lambda$ in the first and $r \geq 4$ and $\lambda \geq 45$ in the last inequality. By using a union bound, we find the probability p_{err_3} that at least one $2r$-offspring flips exactly one one-bit and exactly one zero-bit to be at most

$$\begin{aligned} p_{\text{err}_3} &:= \frac{\lambda}{2}\frac{2rk}{n}\left(1-\frac{2r}{n}\right)^{k-1}\frac{2r(n-k)}{n}\left(1-\frac{2r}{n}\right)^{n-k-1} \\ &\leq 2r^2\left(1-\frac{2r}{n}\right)^{n-2} \leq (1+o(1))2e^{2(\ln(r)-r)} < (2+o(1))e^{-\frac{6}{5}r} < 0.017, \end{aligned}$$

for sufficiently large n, using $k/n \leq 1/\lambda$ for the first inequality. The third inequality is due to $\ln x - x \leq -(1-e^{-1})x < -(3/5)x$ for all $x > 0$ and the last inequality stems from $r \geq 4$. The second inequality follows from

$$\left(1-\frac{2r}{n}\right)^{n-2} \leq e^{-\frac{2r}{n}(n-2)} = e^{-2r\left(1-\frac{2}{n}\right)} = (1+o(1))e^{-2r},$$

using again $r \leq \ln\lambda \leq c\ln n$ for some constant c. Let M_r be the number of such offspring. Any other fitness-decreasing flip-combinations of zeroes and ones in the $2r$-subpopulation require an offspring to flip at least two one-bits. The probability that such an offspring is created is at most

$$p_{\text{err}_4} := \frac{\lambda}{2}\binom{k}{2}\left(\frac{2r}{n}\right)^2 \leq \frac{\ln^2\lambda}{16\lambda} < 0.021,$$

using $k \leq n/\lambda$ and $r \leq \ln\lambda/4$ and the fact that $(\ln^2 x)/x$ is decreasing for $x \geq e^2$ and $\lambda \geq 45 > e^2$.

The events $N_{r/2} > \lambda e^{-r/2}/4$, $N_{2r} < \lambda e^{-2r(1-\frac{1}{\lambda})}$, $M_r = 0$, and the event that no fitness-decreasing offspring is created in the $2r$-subpopulation are sufficient to ensure that the best individual is either surely from the $r/2$-population or chosen uniformly at random from the $N_{r/2} + N_{2r}$ offspring. Conditioning on these events, we have that the probability that the best offspring is chosen from the $r/2$-population is at least

$$\frac{N_{r/2}}{N_{r/2}+N_{2r}} \geq \frac{1}{1+4e^{-r\left(2\left(1-\frac{1}{\lambda}\right)-\frac{1}{2}\right)}} > 0.988.$$

Hence, using a union bound for the error probabilities, the unconditional probability is at least

$$\frac{(1-p_{\text{err}_1}-p_{\text{err}_2}-p_{\text{err}_3}-p_{\text{err}_4})\cdot 0.988+\frac{1}{2}}{2} > 0.5099.$$

□

We note that the restriction $r_t \geq 4$ in the lemma above is not strictly necessary. Also for smaller r_t, the probability that the winning individual is chosen from the $r_t/2$-population is by an additive constant larger than $1/2$. Showing this, however, would need additional proof arguments as for smaller r_t, the event that both subpopulations contain individuals with fitness $k-1$ becomes more likely. We avoid this additional technicality by only arguing for $r_t \geq 4$, which is enough since any constant r_t is sufficient for the fitness drift we need (since we do not aim at making the leading constant precise).

In the following proof of the analysis of the near region, we use the above lemma (with quite some additional arguments) to argue that the r-value quickly reaches 4 or less and from then on regularly returns to this region. This allows to argue that in the near region we have a speed-up of a factor of $\Theta(\lambda)$ compared to the (1+1) EA, since every offspring only has a probability of $O(1/\lambda)$ of making progress (see also [11, 22]).

Theorem 8. *Assume $k \leq n/\lambda$ for the current* OneMax*-value of the self-adjusting (1+λ) EA. Then the expected number of generations until the optimum is reached is $O(n\log(n)/\lambda + \log n)$.*

Proof. The aim is to estimate the OneMax-drift at the points in time (generations) t where $r_t = O(1)$. To bound the expected number of generations until the mutation rate has entered this region, we basically consider the stochastic process $Z_t := \max\{0, \lceil \log_2(r_t/c) \rceil\}$, where $c := 4$, which is the lower bound on r_t from Lemma 7. However, as we do not have proved a drift of Z_t towards smaller values in the region $L := (\ln\lambda)/4 \leq r_t \leq 16\ln\lambda =: U$ (where 16 is an upper bound on $c_2(k)$ from Lemma 5), we use the potential function

$$X_t(Z_t) := \begin{cases} Z_t & \text{if } c \leq r_t \leq L \\ \log_2(L/c) + \sum_{i=1}^{\lceil \log_2(r_t/L) \rceil} 4^{-i} & \text{if } L < r_t < U \\ \log_2(L/c) + \sum_{i=1}^{\log_2(U/L)} 4^{-i} + 4^{-\log_2(U/L)} \lceil \log_2(r_t/U) \rceil & \text{otherwise.} \end{cases}$$

assuming that L and U have been rounded down and up to the closest power of 2, respectively.

The potential function has a slope of 1 for $c \leq r_t \leq L$. Lemma 7 gives us the drift $E(X_t - X_{t+1} \mid X_t; c \leq r_t \leq L) \geq 0.5099 - 0.4901 = 0.0198$. The function satisfies $X_t(Z_t) - X_t(Z_t - 1) \geq 4(X_t(Z_t) - X_t(Z_t + 1))$ if $L < r_t < U$, which corresponds to the region where the probability of decreasing Z_t by 1 has only be bounded from below by 1/4 due to the random steps. Still, $E(X_t - X_{t+1} \mid X_t; L < r_t < U) \geq (1/4)4^{\log_2(U/L)+1} - (3/4)4^{\log_2(U/L)} = \Omega(1)$ in this region due to the concavity of the potential function. Finally, $E(X_t - X_{t+1} \mid X_t; r_t \geq U) = 4^{-\log_2(U/L)}\Omega(1) = \Omega(1)$ by Lemma 5. Hence, altogether $E(X_t - X_{t+1} \mid X_t; r_t \geq c) \geq \kappa$ for some constant $\kappa > 0$. As $X_0 = O(\log n)$, additive drift analysis yields an expected number of $O(\log n)$ generations until for the first time $X_t = 0$ holds, corresponding to $r_t \leq c$. We denote this hitting time by T.

We now consider an arbitrary point of time $t \geq T$. The aim is to show a drift on the OneMax-value, depending on the current OneMax-value Y_t, which satisfies $Y_t \leq n/\lambda$

with probability 1. To this end, we will use Lemma 2. We choose b large enough such that $2e^{-2b\cdot\kappa/4} \le 1-\delta$ holds for some positive constant $\delta > 0$ and note that b is constant. We consider two cases for r_t. If $r_t \le c + 2^b$, which happens with probability at least δ, we have $r_t = O(1)$ and obtain a probability of at least

$$1 - \left(1 - \binom{Y_t}{1}\left(\frac{r_t}{n}\right)\left(1 - \frac{r_t}{n}\right)^{n-1}\right)^{\lambda} \ge 1 - \left(1 - \Theta\left(\frac{Y_t}{n}\right)\right)^{\lambda} = \Omega(\lambda Y_t/n)$$

to improve the OneMax-value by 1, using that $Y_t = O(n/\lambda)$. If $r_t > c + 2^b$, we bound the improvement from below by 0. Using the law of total probability, we obtain

$$E(Y_t - Y_{t+1} \mid Y_t; Y_t \le n/\lambda) = \delta\Omega(\lambda Y_t/n) = \Omega(\lambda Y_t/n).$$

Now a straightforward multiplicative drift analysis (Theorem 3 using $\delta = \Theta(\lambda/n)$) gives an expected number of $O((n/\lambda)\log Y_0) = O(n\log(n)/\lambda)$ generations until the optimum is found. Together with the expected number $O(\log n)$ until the r-value becomes at most c, this proves the theorem. □

7 Putting Everything Together

In this section, we put together the analyses of the different regimes to prove our main result.

Proof of Theorem 1. The lower bound actually holds for all unbiased parallel black-box algorithms, as shown in [1].

We add up the bounds on the expected number of generations spent in the three regimes, more precisely we add up the bounds from Theorem 6, Theorem 7 and Theorem 8, which gives us $O(n/\log\lambda + n\log(n)/\lambda + \log n)$ generations. Due to our assumption $\lambda = n^{O(1)}$ the bound is dominated by $O(n/\log\lambda + n\log(n)/\lambda)$ as suggested. □

Proof of Lemma 1. We basically revisit the regions of different OneMax-values analyzed in this paper and bound the time spent in these regions under the assumption $r = \ln(\lambda)/2$. In the far region, Lemmas 4 and 5, applied with this value of r, imply a fitness drift of $\Omega(\log(\lambda)/\log(en/k))$ per generation, so the expected number of generations spent in the far region is $O(n/\log\lambda)$ as computed by variable drift analysis in the proof of Theorem 6.

The middle region is shortened at the lower end. For $k \ge n/\sqrt{\lambda}$, Lemma 6 gives a fitness drift of $\Omega(1)$, implying by additive drift analysis $O(n/\log\lambda)$ generations to reduce the fitness to at most $n/\sqrt{\lambda}$.

In the near region, which now starts at $n/\sqrt{\lambda}$, we have to argue slightly differently. Note that every offspring has a probability of at least $(1-r)^n \ge e^{-\ln(\lambda)/2+O(1)} = \Omega(\lambda^{-1/2})$ of not flipping a zero-bit. Hence, we expect $\Omega(\sqrt{\lambda})$ such offspring. We pessimistically assume that the other individuals do not yield a fitness improvement; conceptually, this reduces the population size to $\Omega(\sqrt{\lambda})$ offspring, all of which are guaranteed not to flip a

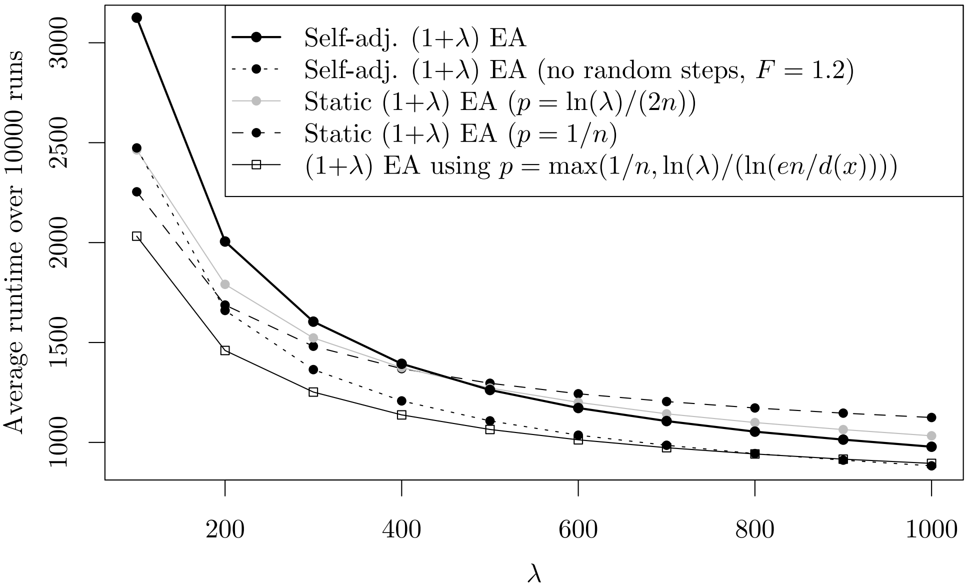

Figure 1: Static and Self-adjusting (1+λ) EA average runtime comparison ($n = 5000$) on OneMax

zero-bit. Adapting the arguments from the proof of Theorem 8, the probability that at least of one of these individuals flips at least a one-bit is at least

$$1 - \left(1 - \binom{Y_t}{1}\left(\frac{r_t}{n}\right)\right)^{\Omega(\sqrt{\lambda})} \geq 1 - \left(1 - \Theta\left(\frac{Y_t}{n}\right)\right)^{\Omega(\sqrt{\lambda})} = \Omega(\sqrt{\lambda}Y_t/n),$$

which is a lower bound on the fitness drift. Using the multiplicative drift analysis, the expected number of generations in the near region is $O(n\log(n)/\sqrt{\lambda})$. Putting the times for the regions together, we obtain the lemma. □

8 Experiments

Since our analysis is asymptotic in nature we performed some elementary experiments in order to see whether besides the asymptotic runtime improvement (showing an improvement for an unspecified large problem size n) we also see an improvement for realistic problem sizes. For this purpose we implemented the (1+λ) EA in C using the GNU Scientific Library (GSL) for the generation of pseudo-random numbers.

The plot in Figure 1 displays the average runtime over 10000 runs of the self-adjusting (1+λ) EA on OneMax for $n = 5000$ as given in Algorithm 1 over $\lambda = 100, 200, \ldots, 1000$. We set the initial mutation rate to 2, i. e., the minimum mutation rate the algorithm can attain. Moreover, the plot displays the average runtime of the classic (1+λ) EA using a static mutation probability of $1/n$.

The average runtimes of both algorithms profit from higher offspring population sizes λ leading to lower average runtimes as λ increases. Interestingly, the classic (1+λ) EA outperforms the self-adjusting (1+λ) EA for small values of λ up to $\lambda = 400$. For higher offspring population sizes the self-adjusting (1+λ) EA outperforms the classic one, indicating that the theoretical performance gain of $\ln \ln \lambda$ can in fact be relevant in practice. Furthermore, we implemented the self-adjusting (1+λ) EA without the random steps, that is, when the rate is always adjusted according to how the best offspring are distributed over the two subpopulations. The experiments show that this variant of the self-adjusting (1+λ) EA performs generally slightly better on OneMax. Since the OneMax fitness landscape is structurally very simple, this result is not totally surprising. It seems very natural that the fitness of the best of $\lambda/2$ individuals, viewed as a function in the rate, is a unimodal function. In this case, the advantage of random steps to be able to leave local optima of this function is not needed. On the other hand, of course, this observation suggests to try to prove our performance bound rigorously also for the case without random rate adjustments. We currently do not see how to do this. Lastly, we implemented the (1+λ) EA using the fitness-depending mutation rate $p = \max\{\frac{\ln \lambda}{n \ln(en/d)}, \frac{1}{n}\}$ as presented in [1]. The experiments suggest that this scheme outperforms all other variants considered.

Additionally we implemented another variant of the self-adjusting (1+λ) EA using three equally-sized subpopulations i. e. the additional one is using (that is, exploiting) the current mutation rate. We compared this variant with the self-adjusting (1+λ) EA, both with and without using random steps. The results are shown in Figure 2. The experiments suggest that the variant using three subpopulations outperforms the self-adjusting (1+λ) EA slightly for small population sizes. For very high population sizes, using just two subpopulations seems to be a better choice.

To gain some understanding on how the parameters influence the runtime, we implemented the self-adjusting (1+λ) EA using different mutation rate update factors, that is, we consider the self-adjusting (1+λ) EA as given in Algorithm 1 where the mutation rate r_t is increased or decreased by some factor F (instead of the choice $F = 2$ made in Algorithm 1). Note that we do not change the rule that we use the rates $r/2$ and $2r$ to create the subpopulations. Furthermore, after initialization, the algorithm starts with rate F and the rate is capped below by F and above by $1/(2F)$ during the run, accordingly.

The results are shown in Figure 3. The plot displays the average runtime over 10000 runs of the self-adjusting (1+λ) EA on OneMax for $n = 5000$ over $\lambda = 100, 200, \ldots, 1000$ using the update factors $F = 2.0, 1.5, 1.01$. The plot suggests that lower values of F yield a better performance. This result is not immediately obvious. Clearly, a large factor F implies that the rate changes a lot from generation to generation (namely by a factor of F). These changes prevent the algorithm from using a very good rate for several iterations in a row. On the other hand, a small value for F implies that it takes longer to adjust the rate to value that is far from the current one.

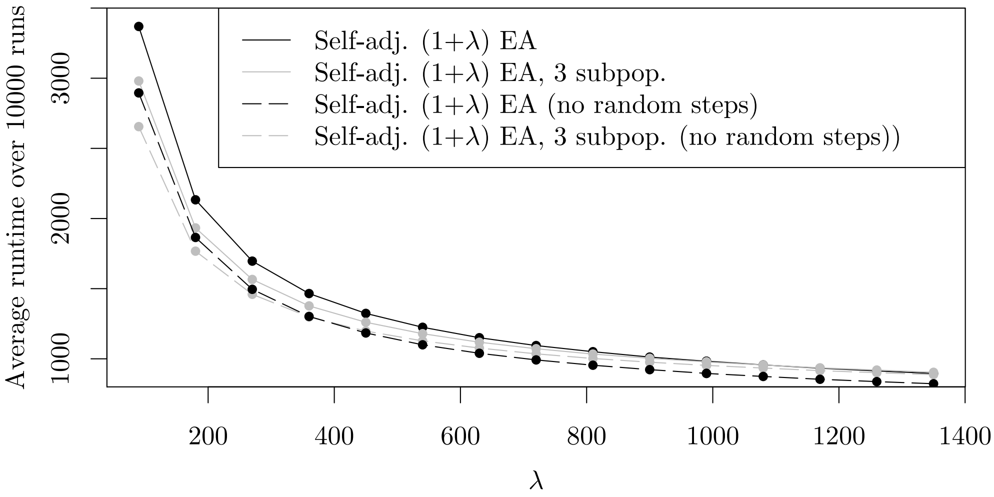

Figure 2: Average runtime of the self-adjusting (1+λ) EA with two and three subpopulations each with and without random steps on OneMax ($n = 5000$)

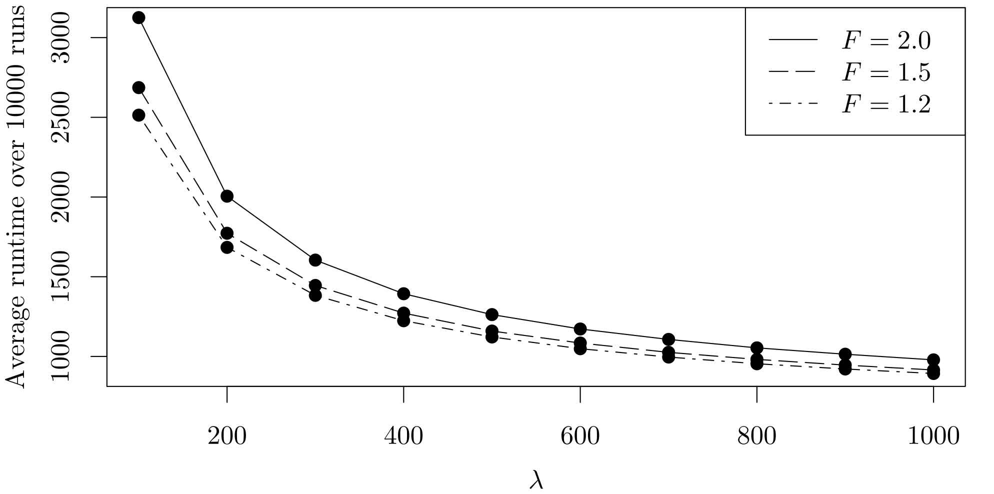

Figure 3: Average runtime of the self-adjusting (1+λ) EA with different mutation rate update factors F on OneMax ($n = 5000$)

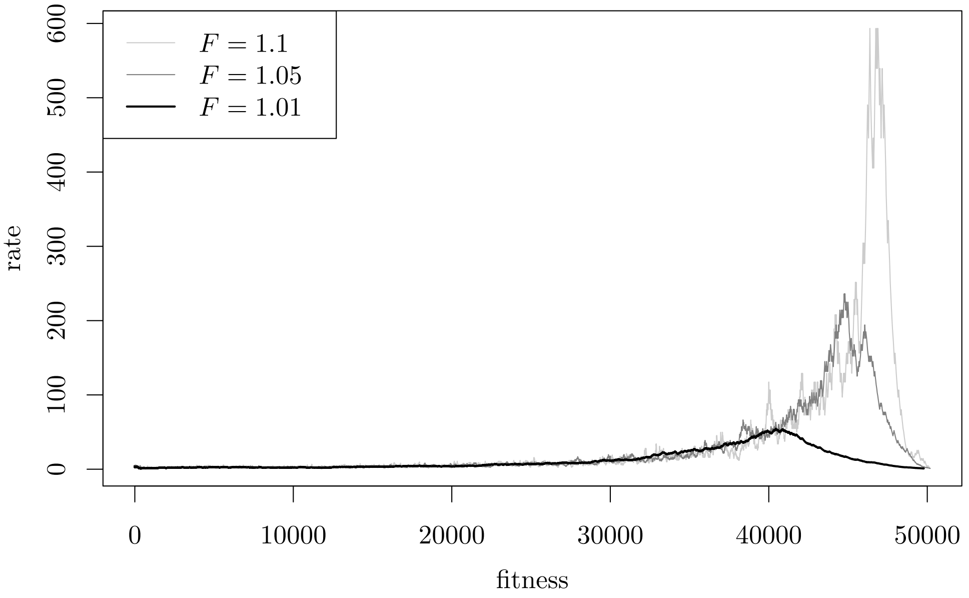

Figure 4: Development of the rate over the fitness of three example runs of the self-adjusting (1+λ) EA on OneMax ($n = 100000$, $\lambda =$ (TODO, I forgot and have to check at home)) using different factors F

Finally, to illustrate the nontrivial development of the rate during a run of the algorithm we plotted the rate of three single runs of the self-adjusting (1+λ) EA using different factors F over the fitness in Figure 4. Since the algorithm initialized with the rate F, the rate increases after initialization decreases again with decreasing fitness-distance to the optimum. The plot suggests that for higher values of F the rate is more unsteady due to the greater impact of the rate adjustments while smaller rate updates yield a more stable development of the rate. Interestingly, for all three values of F, the rates seem to correspond to the same rate after the initial increasing phase from F. Note that this illustration does not indicate the actual runtime. In fact, the specific runtimes are 19766 for $F = 1.01$, 19085 for $F = 1.05$ and 19857 for $F = 1.1$. A similar, more pronounced behaviour can be seen for $F = 2.0$; we chose these particular values of F for illustrative purposes since for $F = 2.0$ the variance in the rate can be visually confusing for the reasons given above.

While we would draw from this experiment the conclusion that a smaller choice of F is preferable in a practical application of our algorithm, the influence of the parameter on the runtime is not very large. So it might not be worth optimizing it and rather view Algorithm 1 as a parameter-less algorithm.

9 Conclusions

We proposed and analyzed a new simple self-adjusting mutation scheme for the (1+λ) EA. It consists of creating half the offspring with a slightly larger and the rest with a slightly smaller mutation rate. Based on the success of the subpopulations, the mutation rate is adjusted. This simple scheme overcomes difficulties of previous self-adjusting choices, e.g., the careful choice of the exploration-exploitation balance and the forgetting rate in the learning scheme of [18].

We proved rigorously that this self-adjusting (1+λ) EA optimizes the OneMax test function in an expected number of $O(n\lambda/\log\lambda + n\log n)$ fitness evaluations. This matches the runtime shown in [1] for a careful fitness-dependent choice of the mutation rate, which was also shown to be asymptotically optimal among all λ-parallel black-box optimization algorithms. Hence our runtime result indicates that the self-adjusting mechanism developed in this work is able to find very good mutation rates. To the best of our knowledge, this is the first time that a self-adjusting choice of the mutation rate speeds up a mutation-based algorithm on the OneMax test function by more than a constant factor.

The main technical challenge in this work is to analyze the quality of the best offspring. In contrast to most previous runtime analyses, where only the asymptotic order of the fitness gain was relevant, we needed a much higher degree of precision as we needed to make statements about in which sub-population the best offspring is, or, in case of multiple best offspring, how they are distributed over the two subpopulations. Note that the quality of the best offspring is not as strongly concentrated around its expectation as, e.g., the average quality.

As a side-result of our analyses, we have observed that using a fixed rate of $r = \ln(\lambda)/2$ gives the bound $O(n/\log\lambda + n\log(n)/\lambda^{1/2})$, which is also asymptotically optimal unless λ is small. However, this setting is far off the usual constant choice of r. It is the first time that a significantly larger mutation rate was shown to be useful in a simple mutation-based algorithm for a simple fitness landscape. Previously, it was only observed that larger mutation rates can be helpful to leave local optima [20].

From this work, a number of open problems arise. A technical challenge is to prove that our algorithm also without the random rate adjustments performs well. This requires an even more precise analysis of the qualities of the offspring in the two subpopulations, for which we currently do not have the methods. From the view-point of understanding the mutation rate for population-based algorithms, two interesting question are (i) to what extent our observation that larger mutation rates are beneficial for the (1+λ) EA on OneMax generalizes to other algorithms and problems, and (ii) for which other problems our self-adjusting choice of the mutation rate gives an improvement over the classic choice of $1/n$ or other static choices.

Acknowledgments This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, and by a grant by the Danish Council for Independent Research (DFF-FNU 4002–00542).

References

[1] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. Unbiased black-box complexity of parallel search. In *Proc. of PPSN '14*, pages 892–901. Springer, 2014.

[2] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proc. of PPSN '10*, pages 1–10. Springer, 2010.

[3] Maxim Buzdalov and Benjamin Doerr. Runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on random satisfiable 3-CNF formulas. In *Proc. of GECCO '17*. ACM, 2017. To appear. Full version available at `http://arxiv.org/abs/1704.04366`.

[4] Stephan Cathabard, Per Kristian Lehre, and Xin Yao. Non-uniform mutation rates for problems with unknown solution lengths. In *Proc. of FOGA '11*, pages 173–180. ACM, 2011.

[5] Jorge Cervantes and Christopher R. Stephens. Rank based variation operators for genetic algorithms. In *Proc. of GECCO '08*, pages 905–912. ACM, 2008.

[6] Duc-Cuong Dang and Per Kristian Lehre. Self-adaptation of mutation rates in non-elitist populations. In *Proc. of PPSN '16*, pages 803–813. Springer, 2016.

[7] Martin Dietzfelbinger, Jonathan E. Rowe, Ingo Wegener, and Philipp Woelfel. Tight bounds for blind search on the integers and the reals. *Combinatorics, Probability & Computing*, 19:711–728, 2010.

[8] Benjamin Doerr. Analyzing randomized search heuristics: tools from probability theory. In Anne Auger and Benjamin Doerr, editors, *Theory of Randomized Search Heuristics*, pages 1–20. World Scientific Publishing, 2011.

[9] Benjamin Doerr. Optimal parameter settings for the $(1+(\lambda,\lambda))$ genetic algorithm. In *Proc. of GECCO '16*, pages 1107–1114. ACM, 2016.

[10] Benjamin Doerr and Carola Doerr. Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings. In *Proc. of GECCO '15*, pages 1335–1342. ACM, 2015.

[11] 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.

[12] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64:673–697, 2012.

[13] Benjamin Doerr, Carola Doerr, and Franziska Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[14] Benjamin Doerr, Carola Doerr, and Timo Kötzing. Solving problems with unknown solution length at (almost) no extra cost. In *Proc. of GECCO '15*, pages 831–838. ACM, 2015.

[15] Benjamin Doerr, Carola Doerr, and Timo Kötzing. The right mutation strength for multi-valued decision variables. In *Proc. of GECCO '16*, pages 1115–1122. ACM, 2016.

[16] Benjamin Doerr, Carola Doerr, and Timo Kötzing. Provably optimal self-adjusting step sizes for multi-valued decision variables. In *Proc. of PPSN '16*, pages 782–791. Springer, 2016.

[17] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. In *Proc. of GECCO '16*, pages 1123–1130. ACM, 2016.

[18] Benjamin Doerr, Carola Doerr, and Jing Yang. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proc. of PPSN '16*, pages 824–834. Springer, 2016.

[19] Benjamin Doerr, Carola Doerr, and Timo Kötzing. Unknown solution length problems with no asymptotically optimal run time. In *Proc. of GECCO '17*. ACM, 2017. To appear.

[20] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Proc. of GECCO '17*. ACM, 2017. To appear. Full version available at `http://arxiv.org/abs/1703.03334`.

[21] Agoston Endre Eiben, Robert Hinterding, and Zbigniew Michalewicz. Parameter control in evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 3:124–141, 1999.

[22] Christian Gießen and Carsten Witt. The interplay of population size and mutation probability in the (1+λ) EA on OneMax. *Algorithmica*, 2017. In press, available online at `http://dx.doi.org/10.1007/s00453-016-0214-z`.

[23] Thomas Jansen and Ingo Wegener. On the analysis of a dynamic evolutionary algorithm. *Journal of Discrete Algorithms*, 4:181–199, 2006.

[24] 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.

[25] Daniel Johannsen. *Random combinatorial structures and randomized search heuristics.* PhD thesis, Saarland University, 2010.

[26] Rob Kaas and Jan M. Buhrman. Mean, median and mode in binomial distributions. *Statistica Neerlandica*, 34:13–18, 1980.

[27] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. (1+1) EA on generalized dynamic OneMax. In *Proc. of FOGA '15*, pages 40–51. ACM, 2015.

[28] Jörg Lässig and Dirk Sudholt. Adaptive population models for offspring populations and parallel evolutionary algorithms. In *Proc. of FOGA '11*, pages 181–192. ACM, 2011.

[29] Per Kristian Lehre and Carsten Witt. Concentrated hitting times of randomized search heuristics with variable drift. In *Proc. of ISAAC '14*, pages 686–697. Springer, 2014.

[30] Boris Mitavskiy, Jonathan E. Rowe, and Chris Cannings. Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *International Journal of Intelligent Computing and Cybernetics*, 2:243–284, 2009.

[31] Frank Neumann and Ingo Wegener. Randomized local search, evolutionary algorithms, and the minimum spanning tree problem. *Theoretical Computer Science*, 378:32–40, 2007.

[32] Pietro Simone Oliveto, Per Kristian Lehre, and Frank Neumann. Theoretical analysis of rank-based mutation - combining exploration and exploitation. In *Proc. of CEC '09*, pages 1455–1462. IEEE, 2009.

[33] Ingo Wegener. Simulated annealing beats Metropolis in combinatorial optimization. In *Proc. of ICALP '05*, pages 589–601. Springer, 2005.

[34] Christine Zarges. Rigorous runtime analysis of inversely fitness proportional mutation rates. In *Proc. of PPSN '08*, pages 112–122. Springer, 2008.

[35] Christine Zarges. On the utility of the population size for inversely fitness proportional mutation rates. In *Proc. of FOGA '09*, pages 39–46. ACM, 2009.