Runtime analysis for self-adaptive mutation rates

Doerr, Benjamin; Witt, Carsten; Yang, Jing

Published in:
2018 Proceedings of the Genetic and Evolutionary Computation Conference

Link to article, DOI:
10.1145/3205455.3205569

Publication date:
2018

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

Runtime Analysis for Self-adaptive Mutation Rates

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

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

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

ABSTRACT

We propose and analyze a self-adaptive version of the $(1, \lambda)$ evolutionary algorithm in which the current mutation rate is part of the individual and thus also subject to mutation. A rigorous runtime analysis on the OneMax benchmark function reveals that a simple local mutation scheme for the rate leads to 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 and $(1 + \lambda)$ EA for all static mutation rates and best possible among all λ-parallel mutation-based unbiased black-box algorithms.

Our result shows that self-adaptation in evolutionary computation can find complex optimal parameter settings on the fly. At the same time, it proves that a relatively complicated self-adjusting scheme for the mutation rate proposed by Doerr et al. (GECCO 2017) can be replaced by our simple endogenous scheme. Moreover, the paper contributes new tools for the analysis of the two-dimensional drift processes arising in self-adaptive EAs, including bounds on occupation probabilities in processes with non-constant drift.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**; *Optimization with randomized search heuristics*;

KEYWORDS

self-adaptive evolutionary algorithms; theory; runtime analysis

ACM Reference Format:
Benjamin Doerr, Carsten Witt, and Jing Yang. 2018. Runtime Analysis for Self-adaptive Mutation Rates. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205569

1 INTRODUCTION

One of the core difficulties when using evolutionary algorithms is finding suitable values for its parameters. It is well known and supported by ample experimental and some theoretical evidence that already small changes of the parameters can have a crucial influence on the efficiency of the algorithm.

GECCO '18, July 15–19, 2018, Kyoto, Japan

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205569

One elegant way to overcome this difficulty, and in addition the difficulty that the optimal parameter values may change during a run of the algorithm, is to let the algorithm optimize the parameters *on the fly*. Formally speaking, this is an even more complicated task, because instead of a single good parameter value now a suitable functional dependence of the parameter on the search history needs to be found. Fortunately, a number of natural heuristics like the 1/5-th rule have proven to be effective in certain cases. In a sense, these are all *exogenous* parameter control mechanisms which are added to the evolutionary system.

An even more elegant way is to incorporate the parameter control mechanism into the evolutionary process, that is, to attach the parameter value to the individual, to modify it via (extended) mutation operators, and to use the fitness-based selection mechanisms of the algorithm to ensure that good parameter values become dominant in the population. This *self-adaptation* of the parameter values has two main advantages: (i) It is generic, that is, the adaption mechanism is provided by the algorithm, only the representation of the parameter in the individual and the extension of the variation operators has to be provided by the user. (ii) It allows to re-use existing algorithms and much of the existing code.

Despite these advantages, self-adaptation is not used a lot in discrete evolutionary optimization. From the theory side, some advice exists how to set up such a self-adaptive system, but a real proof for its usefulness is still missing. This is the point we aim to make some progress on.

Our results: The main result of this work is that we propose a version of the $(1, \lambda)$ evolutionary algorithm (EA) with a natural self-adaptive choice of the mutation rate and prove that it optimizes the classic OneMax benchmark problem in a runtime that is asymptotically optimal among all λ-parallel black-box optimization algorithms and that is, from $\lambda = \Omega((\ln n)^{1+\epsilon})$ on, better than the runtimes of the (1,λ) EA and the (1+λ) EA for all static choices of the mutation rate. It obviously cannot beat the (also asymptotically optimal) runtimes of the (1+λ) EA with fitness-dependent mutation rate of Badkobeh, Lehre, and Sudholt [2] and of the (1+λ) EA with self-adjusting (exogenous) mutation rate of Doerr, Gießen, Witt, and Yang [11]. Hence the good news of our result is that this optimal runtime could be obtained in a generic manner. Note that both the fitness-dependent mutation rate of [2] and the self-adjusting rate of [11] with its mix of random and greedy rate adjustments would have been hard to find without a deeper understanding of the mathematics of these algorithms.

Not surprisingly, the proof of our main result has some similarity to the analysis of the self-adjusting (1+λ) EA of [11]. In particular, we also estimate the expected progress in one iteration and use variable drift analysis. Also, we need a careful probabilistic analysis of the progress obtained from different mutation rates to estimate which rate is encoded in the new parent individual (unfortunately,

we cannot reuse the analysis of [11] since it is not always strong enough for our purposes). The reason, and this is also the main technical challenge in this work, is that the (1,λ) EA can lose fitness in one iteration. This happens almost surely when the mutation rate is too high. For this reason, we need to argue more carefully that such events do not happen regularly. To do so, among several new arguments, we also need a stronger version of the occupation probability result [18, Theorem 7] since (i) we need sharper probability estimates for the case that movements away from the target are highly unlikely and (ii) for our process, the changes per time step cannot be bounded by a small constant. We expect our new result (Lemma 2.3) to find other applications in the theory of evolutionary algorithms in the future. Note that for the (1+λ) EA, an excursion into unfavorable rate regions is less a problem as long as one can show that the mutation rate returns into the good region after a reasonable time. The fact that the (1,λ) EA can lose fitness also makes it more difficult to cut the analysis into regimes defined by fitness levels since it is now possible that the EA returns into a previous regime. In this work, we also gained two insights which might be useful in the design of future self-adaptive algorithms.

Need for non-elitism: Given the previous works, it would be natural to try a self-adaptive version of the (1+λ) EA. However, this is risky. While the self-adjusting EA of [11] can cope with the fact that the current mutation rate is far from the ideal one and in this case provably quickly changes the rate to an efficient setting, a self-adaptive algorithm cannot do so. Since the mutation rate is encoded in the individual, a change of the rate can only occur if an offspring is accepted. For an elitist algorithm like the (1+λ) EA, this is only possible when an offspring is generated that is good enough to compete with the parent(s). Consequently, if the parent individual in a self-adaptive (1+λ) EA has a high fitness, but a detrimental (that is, large) mutation rate, then the algorithm is stuck with this individual for a long time. Already for the simple OneMax function, such a situation can lead to an exponential runtime. Needless to say, when using a comma strategy we have to choose λ sufficiently large to avoid losing the current-best solution too quickly; see [21] for a precise analysis of this phenomenon for the (1,λ) EA using a static mutation rate of $1/n$.

Tie-breaking towards lower mutation rates: To prove our result, we need that the algorithm in case of many offspring of equal fitness prefers those with the smaller mutation rate. Given that the usual recommendation for the mutation rate is small, namely $\frac{1}{n}$, and that it is well-known that large rates can be very detrimental, it is natural to prefer smaller rates in case of ties (where, loosely speaking, the offspring population gives not hint which rate is preferable). This choice is similar to the classic tie-breaking rule of preferring offspring over parents in case of equal fitness (again, the fitness indicates no preference, but the simple fact that one is maybe working already for quite some time with this parent suggest to rather prefer the new individual). Without proof, we remark that without this tie-breaking rule we would need $\lambda = n^{\Omega(1)}$ to ensure a positive drift towards the optimum throughout the process.

Previous works: This being a theoretical paper, for reasons of space we shall mostly review the relevant theory literature, so we only refer to the survey [17] and note that use of self-adaptation in genetic algorithms was proposed in the seminal paper [1] by Bäck. Also, we completely disregard evolutionary optimization in continuous search spaces due to the very different nature of optimization there (visible, e.g., from the fact that dynamic parameter changes, including self-adaptive choices, are very common and in fact necessary to allow the algorithms to approach the optimum with arbitrary precision).

The theoretical analysis of dynamic parameter choices started slow. A first paper [15] on this topic in 2006 demonstrated the theoretical superiority of dynamic parameter choices by giving an artificial example problem for which any static choice of the mutation rate leads to an exponential runtime, whereas a suitable time-dependent choice leads to a polynomial runtime. Four years later [4], it was shown that a fitness-dependent choice of the mutation rate can give a constant-factor speed-up when optimizing the LeadingOnes benchmark function. The first super-constant speed-up on a classic benchmark function obtained from a fitness-dependent parameter choice was shown in [8], soon to be followed by the paper [2] which is highly relevant for this work.

Around that time, several successful self-adjusting ("on the fly") parameter choices were found and analyzed with mathematical means. In [19], Lässig and Sudholt propose a success-based multiplicative update of the population size λ in the (1+λ) EA and show that this can lead to a reduction of the parallel runtime. By a multiplicative update inspired by the 1/5-th success rule from evolution strategies, [7] automatically finds parameter settings leading to the same performance as the fitness-dependent choice in [8]. A learning-based approach was used in [9] to automatically adjust the mutation strength and obtain the performance of the fitness-dependent choice of [10]. Again a different approach was proposed in [11], where the mutation rate for the (1+λ) EA was determined on the fly by creating half the offspring with a smaller and half the offspring with a larger mutation rate than the value currently thought to be optimal. As new mutation rate, with probability $\frac{1}{2}$ the rate which produced the best offspring was chosen, with probability $\frac{1}{2}$ a random of the two rates used was chosen. The three different exogenous approaches used in these works indicate that a generic approach towards self-adjusting parameter choices, such as self-adaptation, would ease the design of such algorithms significantly.

Surprisingly, prior to this work only a single runtime analysis paper for self-adapting parameter choices appeared. In [5], Dang and Lehre show several positive and negative results on the performance of a simple class of self-adapting evolutionary algorithms having the choice between several mutation rates. Among them, they show that such an algorithm having the choice between an appropriate and a destructively high mutation rate can optimize the LeadingOnes benchmark function in the usual quadratic time, whereas the analogous algorithm using a random of the two mutation rates (and hence in half the cases the right rate) fails badly and needs an exponential time. As a second remarkable result, they give an example setting where any constant mutation rate leads to an exponential runtime, whereas the self-adapting algorithm succeeds in polynomial time. As for almost all such examples, also this one is slightly artificial and needs quite some assumptions, for example, that all λ initial individuals are based on the 1-point local optimum. Nevertheless, this result makes clear that self-adaptation can outperform static parameter choices. In the light of this result,

the main value of our results is showing that asymptotic runtime advantages from self-adaptation can also be obtained in less constructed examples (of course, at the price that the runtime gap is not exponential).

This paper is structured as follows. Section 2 defines the self-adaptive (1,λ) EA and provides important mathematical tools used in the analysis. Section 3 presents the main theorem. Its proof considers two main regions of different fitness, which are dealt with in separate subsections. We then finish with some conclusions. Due to space restrictions, the proofs of some lemmas had to be omitted.

2 PRELIMINARIES

2.1 A Self-Adaptive (1,λ) EA

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

To encode the mutation rate into the individual, we extend the individual representation by adding the rate parameter. Hence the extended individuals are pairs (x, r) consisting of a search point $x \in \{0,1\}^n$ and the rate parameter r, which shall indicate that r/n is the mutation rate this individual was created with.

The extended mutation operator first changes the rate to either r/F or Fr with equal probability ($F > 1$). It then performs standard bit mutation with the new rate.

In the selection step, we choose from the offspring population an individual with best fitness. If there are several such individuals, we prefer individuals having the smaller rate r/F, breaking still existing ties randomly. In this winning individual, we replace the rate by F if it was smaller and by $n/(2F)$ if it was larger.

We formulate the algorithm to start with an initial mutation rate r^{init}, which we require to be in $[F, n/(2F)]$. For the result we shall show in this work, the initial rate is not important, but without this prior knowledge we would strongly recommend to start with the smallest possible rate $r^{\text{init}} = F$. Due to the multiplicative rate adaptation, the rate can quickly grow if this is profitable. On the other hand, a too large initial rate might lead to an erratic initial behavior of the algorithm. For the adaptation parameter, we shall use $F = 32$ in our runtime analysis. Having such a large adaptation parameter eases the already technical analysis, because now the two competing rates r/F and Fr are different enough to lead to a significantly different performance. For a practical application, we suspect that a smaller value of F is preferable as it leads to a more stable optimization process. The choice of the offspring population size depends mostly on the degree of parallelism one wants to obtain. Clearly, λ should be at least logarithmic in n to prevent a too quick loss of the current-best solution. For our theoretical analysis, we require $\lambda \geq (\log n)^{1+\varepsilon}$ for an arbitrarily small constant ε. We do not know if the small extra $(\log n)^{\varepsilon}$ factor is necessary, but this is maybe also not the most important question in this area.

Algorithm 1 The (1,λ) EA with self-adapting mutation rate, adaptation parameter $F > 1$, and initial mutation rate r^{init}/n such that $r^{\text{init}} \in [F, n/(2F)]$.

Select x_0 uniformly at random from $\{0,1\}^n$.
Set $r_0 \leftarrow r^{\text{init}}$.
for $t \leftarrow 1, 2, \ldots$ **do**
  for $i \leftarrow 1, \ldots, \lambda$ **do**
    Choose $r_{t,i} \in \{r_{t-1}/F, Fr_{t-1}\}$ uniformly at random.
    Create $x_{t,i}$ by flipping each bit in x independently with probability $r_{t,i}/n$.
  Choose $i \in [1..\lambda]$ such that $f(x_{t,i}) = \min_{j \in [1..\lambda]} f(x_{t,j})$; in case of a tie, prefer an i with $r_{t,i} = r_{t-1}/F$; break remaining ties randomly.
  $(x_t, r_t) \leftarrow (x_{t,i}, r_{t,i})$.
  Replace r_t with $\min\{\max\{F, r_t\}, n/(2F)\}$.

2.2 Runtime Analysis

The main result of this work is a mathematical runtime analysis of the performance of the algorithm proposed above on the classic benchmark function OneMax : $\{0,1\}^n \to \mathbb{R}$ defined by $\text{OneMax}(x) = \sum_{i=1}^n x_i$ for all $x = (x_1, \ldots, x_n) \in \{0,1\}^n$. Since such runtime analyses are by now a well-established way of understanding the performance of evolutionary algorithms, we only briefly give the most important details and refer the reader to the textbook [14].

The aim of runtime analysis is predicting how long an evolutionary algorithm takes to find the optimum or a solution of sufficient quality. As implementation-independent performance measure usually the number of fitness evaluations performed in a run of the algorithm is taken. More precisely, the *optimization time* of an algorithm on some problem is the number of fitness evaluations performed until for the first time an optimal solution is evaluated. Obviously, for a (1,λ) EA, the optimization time is essentially λ times the number of iterations performed until an optimum is generated.

As in classic algorithms analysis, our main goal is an asymptotic understanding of how the optimization time depends on the problems size n. Hence all asymptotic notation in the paper will be with respect to n tending to infinity.

2.3 Probabilistic Tools

In our analysis, we use several standard probabilistic tools including Chernoff bounds. All these can be found in many textbook or the book chapter [6]. We mention the following variance-based Chernoff bound due to Bernstein [3], which is less common in this field (but can be found as well in [6]).

Theorem 2.1. *Let $X_1, \ldots, X_n$ be independent random variables. Let b be such that $E(X_i) - b \leq X_i \leq E(X_i) + b$ for all $i = 1, \ldots, n$. Let $X = \sum_{i=1}^n X_i$. Let $\sigma^2 = \sum_{i=1}^n \text{Var}(X_i) = \text{Var}(X)$. Then for all $\lambda \geq 0$,*

$$\Pr(X \geq E(X) + \lambda) \leq \exp\left(-\frac{\lambda^2}{2(\sigma^2 + \frac{1}{3}b\lambda)}\right),$$

$$\Pr(X \leq E(X) - \lambda) \leq \exp\left(-\frac{\lambda^2}{2(\sigma^2 + \frac{1}{3}b\lambda)}\right).$$

We shall follow the common approach of estimating the expected progress and translating this via so-called drift theorems into an estimate for the expected optimization time. We use the variable drift theorem independently found in [16, 20] in slightly generalized form.

Theorem 2.2 (Variable Drift, Upper Bound). *Given a stochastic process, let $(X_t)_{t\geq 0}$ be a sequence of random variables obtained from mapping the random state at time t to a finite set $S \subseteq \{0\} \cup [x_{\min}, x_{\max}]$, 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 $h(x)\colon [x_{\min}, x_{\max}] \to \mathbb{R}^+$ 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 $\Pr(X_0 = x') > 0$

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

2.4 Occupation Probabilities

To analyze the combined process of fitness and rate in the parent individual, we need a tool that translates a local statement, that is, how the process changes from one time step to the next, into a global statement on the occupation probabilities of the process. Since in our application the local process has a strong drift to the target, Theorem 7 from [18] is too weak. Also, we cannot assume that the process in each step moves at most some constant distance. For that reason, we need the following stronger statement.

Lemma 2.3. *Consider a stochastic process X_t, $t \geq 0$, on $\mathbb{R}$ such that for some $p \leq 1/25$ the transition probabilities for all $t \geq 0$ satisfy $\Pr(X_{t+1} \geq X_t + a \mid X_t > 1) \leq p^{a+1}$ for all $a \geq -1/2$ as well as $\Pr(X_{t+1} - 1 \geq a \mid X_t \leq 1) \leq p^{a+1}$ for all $a \geq 0$. If $X_0 \leq 1$ then for all $t \geq 1$ and $k > 1$ it holds that*

$$\Pr(X_t \geq 1 + k) \leq 11\,(ep)^k\,.$$

Proof. We aim at applying Theorem 2.3 in [12]. To this end, we estimate the moment-generating function of the one-step change $X_{t+1} - X_t$. There are two cases depending on X_t: for $X_t \leq 1$, using the monotonicity of $e^{\lambda(X_{t+1}-1)}$ and the fact that $\Pr(X_{t+1} \geq 1 \mid X_t \leq 1) \leq p$, we obtain

$$D(p,\lambda) := E(e^{\lambda(X_{t+1}-1)} \mid X_t \leq 1) \leq E(e^{\max\{\lambda(X_{t+1}-1),0\}} \mid X_t \leq 1)$$
$$\leq e^0(1-p) + \sum_{a=0}^{\infty} e^{\lambda(a+1)} p^{a+1} \leq 1 + \frac{e^\lambda p}{1 - e^\lambda p},$$

and for $X_t > 1$, using the monotonicity of $e^{\lambda(X_{t+1}-X_t)}$ and the fact that $\Pr(X_{t+1} - X_t \geq -1/2 \mid X_t > 1) \leq p^{1/2}$ we have

$$\rho(p,\lambda) := E(e^{\lambda(X_{t+1}-X_t)} \mid X_t > 1) \leq E(e^{\max\{\lambda(X_{t+1}-X_t),-\frac{1}{2}\}} \mid X_t > 1)$$
$$\leq \frac{1-p^{1/2}}{e^{\lambda/2}} + \sum_{a=0}^{\infty} e^{\lambda a/2} p^{(a+1)/2} = \frac{p^{1/2}-p}{(e^\lambda p)^{1/2}} + \frac{p^{1/2}}{1-(e^\lambda p)^{1/2}}.$$

Using $\lambda := \ln(1/(ep))$ such that $e^\lambda p = 1/e$, we have

$$\rho := \rho(p,\lambda) \leq e^{1/2}(p^{1/2}-p) + \frac{p^{1/2}}{1-e^{-1/2}} \leq \frac{4e^{1/2}}{25} + \frac{1/5}{1-e^{-1/2}} < 0.8,$$

$$D := D(p,\lambda) \leq 1 + (1/e)/(1-1/e) < 2.$$

Theorem 2.3, inequality (2.8) in [12] yields with $a := 1$ and $b := 1+k$ that

$$\Pr(X_t \geq 1 + k \mid X_0) \leq \rho^t e^{-\lambda(1+k-X_0)} + \frac{1}{1-\rho} D e^{-\lambda k}$$
$$\leq (ep)^k + \frac{2}{0.2}(ep)^k = 11(ep)^k.$$

□

3 MAIN RESULT AND PROOF

Our main result is as follows.

Theorem 3.1. *Let $\lambda \geq (\ln n)^{1+\epsilon}$ for an arbitrary constant $\epsilon > 0$ and $F = 32$. Then the expected number of generations of the self-adapting (1,λ) EA to optimize OneMax is $O(n/\log\lambda + (n\log n)/\lambda)$, corresponding to an expected number of fitness evaluations of $O(n\lambda/\log\lambda + n\log n)$.*

The proof of the theorem is based on a careful, technically demanding drift analysis of both the current OneMax-value k_t (which is also the fitness distance, recall that our goal is the minimization of the objective function) and the current rate r_t of the parent. In very rough terms, a similar division of the run as in [11] into regions of large OneMax-value, the far region (Section 3.1), and of small OneMax-value, the near region (Section 3.2) is made. The middle region considered in [11] is subsumed under the far region here.

3.1 The Far Region

In this section, we analyze the optimization behavior of our self-adaptive (1,λ) EA in the regime where the fitness distance k is at least n/λ. Due to our assumption $\lambda \geq (\ln n)^{1+\epsilon}$, it is very likely to have at least one copy of the parent among λ offspring for $r = O(\ln\lambda)$. Thus the (1,λ) EA works almost the same as the $(1+\lambda)$ EA but can lose fitness with small probability. The following lemma is crucial in order to analyze the drift of the rate depending on k, which follows a similar scheme as with the $(1+\lambda)$ EA proposed in [11].

Roughly speaking, the rate leading to optimal fitness progress is n for $k \geq n/2 + \omega(\sqrt{n\ln(\lambda)})$, $n/2$ for $k = n/2 \pm o(\sqrt{n\log(\lambda)})$, and then the optimal rate quickly drops to $r = \Theta(\log\lambda)$ when $k \leq n/2 - \varepsilon n$.

To ease the representation, we first define two fitness dependent bound $L(k)$ and $R(k)$.

Definition 3.2. Let n be sufficiently large, $n/\ln\lambda < k < n/2$ and $F = 32$. We define $L(k) := (F\ln(en/k))^{-1}$ and $U(k) := n(2n-k)/(20(n-2k)^2)$.

According to the definition, both $L(k)$ and $R(k)$ monotonically increase when k increases.

Lemma 3.3. *Consider an iteration of the self-adaptive (1,λ) EA with current fitness distance k and current rate r. Let $F = 32$ and n sufficiently large. Then:*

- *If $n/\ln\lambda < k$ and $F \leq r \leq L(k)\ln\lambda$, the probability that all best offspring have been created with rate Fr is at least $1 - O(\ln^3(\lambda)/\lambda^{1/(4\ln\ln\lambda)})$.*
- *If $k < n/2$ and $n/(2F) \geq r \geq U(k)\ln\lambda$, then the probability that all best offspring have been created with rate r/F is at least $1 - \lambda^{1-1.15r/(U(k)\ln\lambda)}$.*

Lemma 3.3 will be crucial in order to bound the expected progress on fitness in the far region. We notice that in the lemma we may allow $r > \ln\lambda$ when k is large and $r = \Theta(n)$ when $k = n/2 - \Theta(\sqrt{n\ln\lambda})$. It is easy to show a positive progress on fitness for $r < \ln\lambda$ since there will be sufficiently many offspring that do not

flip zeroes. When $r \geq \ln \lambda$ we expect all offspring to flip zeros, but we can still show a positive drift when $k > 7n/20$, as stated in the following lemma. The idea is that the standard variation of the number of flipping ones is $\sqrt{kr/n(1-r/n)} = \Theta(\sqrt{r})$. This makes a deviation compensating bad flips among the remaining $n-2k$ zeros likely enough.

Lemma 3.4. *Let $7n/20 \leq k < n/2$ and n be large enough. Let $F = 32$ and $\alpha = 10^{-4}$. Assume $r \leq \min\{1.01n^2 \ln \lambda/(11.95(n-2k)^2), n/(2F)\}$. Assume that from a parent with fitness distance k we generate an offspring using standard bit mutation with mutation rate $p = r/n$. Then the probability that this offspring has a fitness distance of at most $k - s$ with $s := \alpha(\min\{\ln \lambda, r\} + (n-2k)r/n)$, is at least $\lambda^{-0.98}/20$.*

For $k < 7n/20$, we need a more careful analysis, where we will estimate the expected progress on fitness averaged over the random rates the algorithm may have at a time. Hence, we assume a fixed current fitness but a random current rate and compute the average drift of fitness with respect to the distribution on the rates. This approach is similar to the one by Jägersküpper [13], who computes the average drift of the Hamming distance to the optimum when the (1+1) EA is optimizing a linear function, where the average is taken with respect to a distribution on all search points with a certain Hamming distance.

Of course, we want to exploit that a rate yielding near-optimal fitness progress is used most of the time such that too high (or too low) rates do not have a significant impact. To this end, Lemma 2.3 about occupation probabilities will be crucial.

We now define two fitness dependent bounds $r_l(k)$ and $r_u(k)$. We show in Lemma 3.6 that for any rate, if r/F or Fr is within the bounds, then the algorithm has logarithmic drift on fitness.

Definition 3.5. Let n be sufficiently large, $n/\ln \lambda < k < n/2$ and $F = 32$. We define

$$r_u(k) := \begin{cases} n^2 \ln(\lambda)/(11.95(n-2k)^2) & \text{if} \quad 7n/20 \leq k < n/2, \\ 80U(k)\ln(\lambda)/79 & \text{if} \quad n/\ln\lambda < k < 7n/20. \end{cases}$$

$$r_l(k) := \begin{cases} L(k)\ln(\lambda)/2 & \text{if} \quad n/\ln\lambda \leq k < n/2, \\ F & \text{if} \quad n/\lambda < k < n/\ln\lambda. \end{cases}$$

We notice that Lemma 3.3 can be applied to all $r > r_u$ or $r < r_l$ because for all $7n/20 \leq k < n/2$, we have $r_u/(U(k)\ln\lambda) = 20n/(11.95(2n-k)) \geq 20/(11.95(2-0.35)) \geq 80/79$. For $k < n/\ln\lambda$, we set r_l to the minimal possible value of r. Finally note that r_u is non-decreasing in k due to the monotonicity of $n^2/(n-2k)^2$ and $U(k)$.

Lemma 3.6. *Let n be sufficiently large, $n/\lambda < k < n/2$ with $F = 32$. Suppose that $\lambda \geq c \ln n$ for some sufficiently large constant $c > 0$ and $\lambda = n^{O(1)}$. Let $\Delta(k,r)$ denote the fitness gain of the best offspring.*

- *If $r \geq 1.01Fr_u$, then $E(\Delta(k,r)) \geq -(1+o(1))(n-2k)r/(Fn)$.*
- *If $r \leq 1.01Fr_u$ and $k \geq 7n/20$, then $E(\Delta(k,r)) \geq (1-o(1))10^{-4}((n-2k)r/(Fn) + \min\{\ln\lambda, r/F\})$.*
- *If $r \leq 1.01Fr_u$ and $n/\lambda < k < 7n/20$, then $E(\Delta(k,r)) \geq (1-o(1))\min\{r, \ln(\lambda)/\ln(en/k)\}/F$.*

Proof. The probability of using rate r/F is $1/2$. Thus with probability at least $1-(1/2)^\lambda = 1-o(1/n^3)$, at least one offspring uses rate r/F. For this offspring, the expected loss is $(n-2k)r/(Fn)$. If the complementary event (hereinafter called failure) of probability $o(1/n^3)$ happens, we estimate $\Delta(k,r)$ pessimistically by $-n$. Hence, the contribution of failure events is $o(1/n^2)$ which is lower order of our desired expectation. We thus ignore $\Delta(k,r) < -(n-2k)Fr/n$. This proves the first statement.

To prove the second item, we take $i = 10^{-4}((n-2k)r/(Fn) + \min\{\ln\lambda, r/F\}) = \Omega((n-2k)r/n+1)$ and consider the probability that an offspring uses rate r/F and achieves progress i or more. Applying Lemma 3.4, we obtain $\Pr(\Delta(k,r) > i) \geq 1-(1-\lambda^{-0.98}/40)^\lambda = 1-O(\exp(-\lambda^{0.02}/40)) = 1-o(1)$. If the complementary (failure) event happens, we estimate $\Delta(k,r)$ pessimistically by $-(n-2k)r/(Fn)$. Since $i = \Omega((n-2k)r/n+1)$ and the failure probability is $o(1)$, the contribution of failure events is $o(i)$. Thus the statement holds.

For the third item, we still consider the progress of rate r/F. Notice that for $k < 7n/20$ we have $1.01r_u(k) < 1.01r_u(7n/20) < 0.95\ln\lambda$. Take $i := \min\{r, \ln\lambda/\ln(en/k)\}/F$. The probability that one offspring using rate $r/F < 0.95\ln\lambda$ makes a progress of at least i is lower bounded by

$$\binom{k}{i}\left(\frac{r}{Fn}\right)^i\left(1-\frac{r}{Fn}\right)^n \geq \left(\frac{k}{i}\cdot\frac{r}{Fn}\right)^i\left((1-o(1))e^{-\frac{r}{F}}\right)$$
$$\geq \left(\frac{k}{en}\right)^i e^{-0.95\ln\lambda} \geq \lambda^{-1/F-0.95} > \lambda^{-0.99}.$$

Thus we obtain $\Pr(\Delta(k,r) > i) \geq 1-(1-\lambda^{-0.99}/2)^\lambda = 1-O(\exp(-\lambda^{0.01}/2)) = 1-o(1/\ln\lambda)$. If the failure event happens we estimate $\Delta(k,r)$ pessimistically by $-(n-2k)r/(Fn) = O(\ln\lambda)$. The contribution of failure events is $o(1)$ which is also $o(i)$. Therefore the third statement holds. □

As discussed, our aim is to show that r_t/F or Fr_t stays in the right range frequently enough such that the overall average drift is still logarithmic. We notice that small rates $r_t < r_l$ intuitively do not have a negative effect, therefore we focus on the probability that $r_t < Fr_u$. Since r_u monotonically decreases when k decreases, we need to analyze whether r still stays in the right range if there are large jumps in fitness distance k. Intuitively, the speed at which the mutation rate is decreased is much higher than than the decrease of fitness distance. To make this rigorous, we first look at the probability of large jumps, as detailed in the following lemma.

Lemma 3.7. *Let n be sufficiently large, $n/\ln\lambda < k < n/2$ and $F = 32$. Let $Z(k,r)$ denote the fitness-distance increase when applying standard bit mutation with probability $p = r/n$ to an individual with k ones. Then*

$$\Pr\left(Z(k,r) \leq (n-2k)r/n - \Delta\right) \leq \exp\left(\frac{-\Delta^2}{2(1-p)(r+\Delta/3)}\right),$$
$$\Pr\left(Z(k,r) \geq (n-2k)r/n + \Delta\right) \leq \exp\left(\frac{-\Delta^2}{2(1-p)(r+\Delta/3)}\right).$$

We now use Lemma 3.7 to show that once $r_t \geq Fr_u(k_t)$, there will be a strong drift for $r_t/r_u(k_t)$ to decrease down to 1.

Lemma 3.8. *Let $X_t = \log_F(r_t/r_u(k_t))$. If $k_t < n/2$, we have*

$$\Pr(X_{t+1} - X_t \geq a \mid X_t > 1) \leq \lambda^{-\Omega(a+1)} \text{ for all } a \geq -1/2,$$
$$\Pr(X_{t+1} - 1 \geq a \mid X_t \leq 1) \leq \lambda^{-\Omega(a+1)} \text{ for all } a > 0.$$

Proof. We first notice that $r_{t+1} \in \{Fr_t, r_t/F\}$. If $r_u(k_t) \le r_u(k_{t+1})$ then $X_{t+1} - X_t \le 1$, otherwise if $r_u(k_t) > r_u(k_{t+1})$ then $X_{t+1} - X_t > 1$ is possible. Our intuition is that X_t concentrates below or equal to 1 which means r_t is a useful rate. Let $\tau := \log_F(79/80) < 0$. We take this value because $X_t > \tau$ implies $r_t > 79 r_u(k_t)/80 \ge U(k_t)\ln\lambda$. Thus we can apply Lemma 3.3 for $X_t > \tau$. Since $\Pr(X_{t+1} - 1 \ge a \mid \tau < X_t \le 1) < \Pr(X_{t+1} - X_t \ge a \mid \tau < X_t \le 1)$, it is sufficient to analyze $\Pr(X_{t+1} - X_t \ge a \mid X_t > \tau)$ and $\Pr(X_{t+1} - 1 \ge a \mid X_t \le \tau)$.

If $k_{t+1} \ge k_t$ then $r_u(k_{t+1}) \ge r_u(k_t)$. When $r_{t+1} = r_t/F$ we have $X_{t+1} - X_t \le -1$. Otherwise if $r_{t+1} = Fr_t$ using the fact that $r_u(k_t) > U(k_t)\ln\lambda$ and applying Lemma 3.3 we obtain $\Pr(X_{t+1} - X_t \ge a, k_{t+1} \ge k_t \mid X_t > \tau) \le \Pr(r_{t+1} = Fr_t \mid X_t > \tau) < \lambda^{-0.15}$ for all $a > -1$. Furthermore $\Pr(X_{t+1} - X_t > 1, k_{t+1} \ge k_t) = 0$ and $\Pr(X_{t+1} - 1 \ge 0, k_{t+1} \ge k_t \mid X_t < \tau) = 0$.

If $k_{t+1} < k_t$ then $r_u(k_{t+1}) < r_u(k_t)$. Thus $X_{t+1} - X_t$ depends on how much r_u decreases. Let $\sigma_t^2 := r_u(k_t)/r_u(k_{t+1}) = (n - 2k_{t+1})^2/(n - 2k_t)^2$. Then $\sigma > 1$ if and only if $k_{t+1} < k_t$. Using the fact that $r_{t+1} \in \{Fr_t, r_t/F\}$ we bound

$$\begin{aligned}&\Pr\left(X_{t+1} - X_t \ge a, k_{t+1} < k_t \mid X_t > \tau\right)\\ &\le \Pr\left(r_t = r_{t+1}/F, \sigma_t^2 \ge F^{a+1} \mid X_t > \tau\right)\\ &\quad + \Pr\left(r_{t+1} = Fr_t, \sigma_t^2 \ge F^{a-1} \mid X_t > \tau\right).\end{aligned}$$

For $-1/2 \le a < 4$, we apply Lemma 3.3 to obtain

$$\begin{aligned}&\Pr\left(X_{t+1} - X_t \ge a, k_{t+1} < k_t \mid X_t > \tau\right)\\ &\le \Pr\left(\sigma_t^2 \ge F^{a+1} \mid X_t > \tau\right) + \lambda^{-0.15}.\end{aligned}$$

Otherwise for $a \ge 4$,

$$\Pr\left(X_{t+1} - X_t \ge a, k_{t+1} < k_t \mid X_t > \tau\right) \le \Pr\left(\sigma_t^2 \ge F^{a-1} \mid X_t > \tau\right).$$

When we analyze $X_{t+1} - 1 \ge a$ for $a > 0$, we bound

$$\begin{aligned}&\Pr\left(X_{t+1} - 1 \ge a, k_{t+1} < k_t \mid X_t \le \tau\right)\\ &\le \Pr\left(r_t = r_{t+1}/F, \sigma_t^2 \ge F^{a+2-X_t} \mid X_t \le \tau\right)\\ &\quad + \Pr\left(r_{t+1} = Fr_t, \sigma_t^2 \ge F^{a-X_t} \mid X_t \le \tau\right)\\ &\le \Pr\left(\sigma_t^2 \ge F^{a-X_t} \mid X_t \le \tau\right).\end{aligned}$$

It remains to estimate $\Pr(\sigma_t \ge 1+s \mid X_t)$ in three cases respectively: (i) $(1+s)^2 = F^{a-X_t}$ when $X_t \le \tau < 0, a \ge 0$; (ii) $(1+s)^2 = F^{a-1}$ when $X_t > \tau, a > 4$; (iii) $(1+s)^2 = F^{a+1}$ when $X_t > \tau, -1/2 \le a < 4$. The minimal value of s is attained at $(1+s)^2 = F^{-\tau} = 80/79$ when $X_t = \tau$ and $a = 0$. In each case, s increases faster than a. Thus $s = \Omega(1+a)$. We rewrite

$$\begin{aligned}\Pr\left(\sigma_t \ge 1+s \mid X_t\right) &= \Pr\left(\frac{n - 2k_{t+1}}{n - 2k_t} \ge 1+s \mid X_t\right)\\ &= \Pr\left(k_t - k_{t+1} \ge s(n-2k_t)/2 \mid X_t\right).\end{aligned}$$

Let $\Delta_{\hat r} := (s/2 + \hat r/n)(n - 2k_t)$ for $\hat r \in \{r_t/F, Fr_t\}$. Applying Lemma 3.7 and using a union bound we obtain

$$\Pr\left(\sigma_t \ge 1+s \mid X_t\right) < \lambda \exp\left(\max\left\{\frac{-\Delta_{\hat r}^2}{2(1-\hat r/n)(\hat r + \Delta_{\hat r}/3)}\right\}\right).$$

We first show that we only need to consider $(n-2k_t)^2 > 170n\ln(\lambda)$. Because otherwise $r_u(k_t) \ge n/2040 > n/(2F^2)$ results in $X_t = \log_F(r_t/r_u(k_t)) < 1$ even for the maximal possible rate $n/(2F)$. Therefore we consider $\Delta_{\hat r} = \Omega(\sqrt{n\ln\lambda})$. If $r_t = O(\Delta_{\hat r})$ then $\Pr(\sigma_t \ge 1+s \mid X_t \ge 0) < \lambda\exp(-\Theta(\Delta_{\hat r})) = o(1/n^2)$. Otherwise if $r_t = \omega(\Delta_{\hat r})$ we regard $(-(n-2k_t)^2/n)\cdot(s/2+\hat p)^2/(2(1-\hat p)(\hat p + o(1)))$ with $\hat p = \hat r/n$. The minimal value for $(s/2+p)^2/((1-p)p)$ is $s(2+s)$ attained at $p = s/(2(s+1))$. Using $(n-2k_t)^2/n > 170\ln\lambda$ and $s \ge \sqrt{80/79} - 1 > 1/160$, we obtain $\Pr(\sigma_t \ge 1+s) \le \lambda^{-\Omega(s)}$. □

We finally use Lemma 3.8 and Lemma 2.3 to obtain a logarithmic drift on average. After this major effort, it is a matter of a relatively straightforward drift analysis of fitness distance to obtain the following bound on the time to leave the far region.

Theorem 3.9. *For any initial search point k_0 and r_0, the hitting time T for $k_t \le n/\lambda$ has expectation $E(T) = O(n/\log\lambda)$. Moreover with probability at least $1 - o(n^{-1.2})$, it holds $k_{t'} \le 2n/\lambda$ and $r_{t'} < 0.6\ln\lambda$ for some $t' = O(n/\log\lambda)$.*

Proof. We first argue that within an expected number of $O(n/\log\lambda)$ generations we will have $k_t < n/2$. When $k_t > n/2$ any rate between 1 and $n/2$ makes positive progress in expectation. Thus $k_{t+1} < k_t$ with probability at least $1 - \exp(-\Theta(\lambda)) = 1 - o(1/n^2)$. Moreover, if $k_0 - n/2 > \sqrt{n\ln\lambda}$, the drift for k_t towards 0 is $\Omega(\ln\lambda)$ no matter how small r_0 is. This is because according to Lemma 3.3 if $k_t > n/2$ and $r_t = o(\ln\lambda)$ it takes $O(\ln\ln(\lambda))$ iterations in expectation for r_t to increase to $\Omega(\ln\lambda)$. Moreover, once $r_t = \Omega(\ln\lambda)$, we have $k_t - k_{t+1} = \Omega(\ln\lambda)$. Hence, it takes an expected number of $O(n/\ln\lambda)$ generations to obtain a fitness of $k_t < n/2$. Once we have $k_t < n/2$ the probability that $k_{t+1} > n/2$ is less than $\exp(-\Theta(\lambda)) = o(1/n^2)$ since we take the best among λ offspring. We now assume $k_t < n/2$ for all t. We notice that when $0 < (n-2k_t)^2 < 170n\ln\lambda$, any r_t satisfies $r_t < Fr_u(k_t)$. Otherwise, according to Lemma 3.8, as long as $r_t \ge Fr_u(k_t)$, there is strong drift for $r_t/r_u(k_t)$ to decrease. It takes at most $O(\ln n)$ iterations until $r_t < Fr_u(k_t)$ in expectation. The fitness distance k_t may increase during adaptation, but $k_t < n/2$ always holds. Thus, we assume $k_0 < n/2$ and $r_0 < Fr_u(k_0)$ without loss of generality.

The idea of the remaining proof is to compute an average drift for any fixed distance using the distribution of mutation rates, and then to apply the variable drift theorem to obtain a runtime bound. Applying Lemma 3.8 and Lemma 2.3 to the random variables $\log_F(r_t/r_u(k_t))$ we see that $\Pr(r_t \ge F^{1+a} r_u(k_t)) \le \lambda^{-\Omega(a)}$. For distance k, let $r^{(i)}, i \in \mathbb{Z}$, denote the rate between $(1.01F^{i-1}r_u(k), 1.01F^i r_u(k)]$. Therefore for any $i \ge 1$ we obtain $\Pr(r_t = r^{(i+1)} \mid k_t) = o(\Pr(r_t = r^{(i)} \mid k_t))$. According to Lemma 3.6, the average drift satisfies $E(\Delta(k, r^{(i)})) \ge \Omega(-(n-2k)r^{(i)}/n - 2)$ for $i \ge 2$ and $E(\Delta(k, r^{(1)})) \ge \Omega((n-2k)r^{(1)}/n)$. Thus $E(\Delta(k, r^{(i)}))\Pr(r^{(i)}) = o(E(\Delta(k, r^{(i-1)}))\Pr(r^{(i-1)}))$ for $i \ge 2$ and $E(\Delta(k, r^{(2)}))\Pr(r^{(2)}) = o(E(\Delta(k, r^{(1)}))\Pr(r^{(1)}))$. Let $\Delta^{(k)}$ denote the average drift at distance k. We obtain

$$\Delta^{(k)} = \sum_{i\in\mathbb{Z}} E(\Delta(k, r^{(i)}))\Pr(r^{(i)}) \ge \sum_{i<0} E(\Delta(k, r^{(i)}))\Pr(r^{(i)}).$$

We notice that for all $i < 0$ the drift $E(\Delta(k, r^{(i)}))$ is positive. Furthermore due to the result of Lemma 3.3 and the definition of $r_l(k)$

we will have $\Pr(r \geq r_l(k) \mid k) = \Theta(1)$. Therefore the average drift is

$$\begin{aligned}\Delta^{(k)} &\geq \min_{r_l(k) \leq r < r_u(k)} \{E(\Delta(k,r))\} \Pr(r_l(k) \leq r < r_u(k)) \\ &= \Theta(\ln(\lambda)/\ln(n/k)).\end{aligned}$$

Using the variable drift theorem (Theorem 2.2) and the fact that

$$\int_{n/\lambda}^{n/2} \frac{\ln(n/k)}{\ln(\lambda)} \, \mathrm{d}k = \frac{(k\ln(n) - k\ln(k) + k)\big|_{n/\lambda}^{n/2}}{\ln \lambda} = \frac{\Theta(n)}{\ln \lambda},$$

the expected time to reduce the fitness distance to at most n/λ is then $\Theta(n/\ln \lambda)$.

The proof of the second statement is omitted for reasons of space. □

3.2 The Near Region

We now analyze the drift in rate and fitness distance in the regime $k_t = O(n/\lambda)$, the so-called near region. Informally, this region is responsible for the $(n \log n)/\lambda$ term in the expected number of generations since here the probability of an improvement is only $O(1/\lambda)$ and the offspring population can boost this probability by a factor of $\Theta(\lambda)$, assuming constant rate.

We start with a preparatory lemma about the probability of making progress and, in any case, not losing fitness in this regime.

Lemma 3.10. *Let n be sufficiently large, $0 < k \leq 3n/\lambda$ and consider rate r with $r = O(\ln \lambda)$. Let $p_-(r)$ and $p_0(r)$ denote the probability of a single offspring being better than or same as its parent. Then*

$$(1 - o(1))e^{-r} < p_0(r) < (1 + o(1))e^{-r},$$
$$(1 - o(1))\frac{kre^{-r}}{n} < p_-(r) < (1 + o(1))\frac{kre^{-r}}{n}.$$

Proof. We look at the number X of flips in k one-bits and the number Y of flips in $(n-k)$ zero-bits. Then $p_-(r)$ is lower bounded by $p_-(r) > \Pr(X = 1, Y = 0) = \frac{kr}{n}\left(1 - \frac{r}{n}\right)^{n-1} > (1-o(1))\frac{kre^{-r}}{n}$. Using the fact that $kr/n = o(1)$, $kr^2/n = o(1)$ and $(kr^2/n)^{1.5} = o(kr/n)$, $p_-(r)$ is upper bounded by

$$\begin{aligned}p_-(r) &< \Pr(X \in \{1,2\}, Y = 0) + \sum_{i=3}^{2k-1} \Pr(X + Y = i, X > Y) \\ &< \frac{kr}{n}\left(1 - \frac{r}{n}\right)^{n-1} + \frac{k^2r^2}{n^2}\left(1 - \frac{r}{n}\right)^{n-2} \\ &+ \sum_{i=3}^{2k-1} (i-1)\left(\frac{r}{n}\right)^i \left(1 - \frac{r}{n}\right)^{n-i} \binom{k}{\lceil i/2 \rceil}\binom{n-k}{\lfloor i/2 \rfloor} \\ &< \frac{kr}{n}\left(1 - \frac{r}{n}\right)^{n-2}\left(1 - \frac{r}{n} + \frac{kr}{n}\right) + \sum_{i=3}^{2k-1}\left(\frac{r}{n}\right)^i \left(1 - \frac{r}{n}\right)^{n-i} (kn)^{i/2} \\ &< (1 + o(1))\frac{kr}{n}\left(1 - \frac{r}{n}\right)^n + \sum_{i=3}^{2k-1}\left(\frac{kr^2}{n}\right)^{i/2}\left(1 - \frac{r}{n}\right)^{n-i} \\ &< (1 + o(1))\frac{kre^{-r}}{n}.\end{aligned}$$

Similarly for $p_0(r)$ we have

$$p_0(r) > \Pr(X = Y = 0) = \left(1 - \frac{r}{n}\right)^n > (1 - o(1))e^{-r}.$$

Using the fact that $r^2 = o(k/n)$ then

$$\begin{aligned}p_0(r) &= \Pr(X = Y = 0) + \sum_{i=1}^{k} \Pr(X = Y = i) \\ &= \left(1 - \frac{r}{n}\right)^n + \sum_{i=1}^{k} \binom{k}{i}\binom{n-k}{i}\left(\frac{r}{n}\right)^{2i}\left(1 - \frac{r}{n}\right)^{n-2i} \\ &< e^{-r} + \sum_{i=1}^{k}\left(\frac{kr^2}{n}\right)^i e^{-r} < (1 + o(1))e^{-r}.\end{aligned}$$ □

The following lemma is the counterpart of Lemma 3.3 in the near region, where the optimal rate is the smallest possible value F. We observe that now the probability of making a rate-increasing step is no longer bounded by $o(1)$ in general. If $k_t = \Theta(n/\lambda)$ and $r_t = O(1)$, we still have a small constant probability of increasing the rate, which is due to the fact that with constant probability all offspring copy the parent. If $k_t = o(n/\lambda)$, the effect of the tie-breaking rule leads to the second bound that depends on k.

Lemma 3.11. *Let n be sufficiently large, $0 < k \leq 3n/\lambda$ and $F = 32$. The probability that a best offspring has been created with rate Fr is at most $\exp(-9r)$ for $r \geq \ln \lambda$ and $\lambda k e^{-(F-1)r}/n$ for $r < \ln \lambda$.*

The previous lemma is used to bound the average drift of fitness distance in the near region, as detailed by the following lemma. Also a bound on the expected loss in a step is given, which will be used later to estimate the probability that the near region is not left towards the far region again.

Lemma 3.12. *Suppose that $\lambda \geq (\ln n)^{1+\epsilon}$ for some arbitrarily small constant $\epsilon > 0$ and $\lambda = n^{O(1)}$. Let $r_0 = F$ and consider an arbitrary $t \geq 0$. If $k_s \leq 3n/\lambda$ for all $s \in \{0, \ldots, t\}$ then*

(1) $E(k_t - k_{t+1} \mid k_t) = \Omega(\lambda k_t/n)$.
(2) $E((k_{t+1} - k_t) \cdot \mathbb{1}\{k_{t+1} \geq k_t\} \mid k_t) = o(k_t/n)$

The previous lemma can be applied in a relatively straightforward drift analysis on fitness distance to bound the time to cross the near region, resulting in the following theorem.

Theorem 3.13. *Assume $k_0 \leq 2n/\lambda$ and $r_0 \leq (7/9)\ln \lambda$. With probability at least $1/2$, it holds $k_t = 0$ for some $t = O(n \ln(n/\lambda)/\lambda)$.*

Proof. We assume that $k_s \leq 3n/\lambda$ for all $s \in \{0, \ldots, t\}$ and show later that the assumption holds with probability $\Omega(1)$. Under the assumption, we may apply Lemma 3.11.

The first aim is to show that the rate quickly drops to F and then with high probability stays close to this minimum value. The probability of observing $R^* := \log_F((7/9)\ln \lambda)$ rate-decreasing steps in a row is by Lemma 3.11 at least $\prod_{i=0}^{R^*-1}\left(1 - 2e^{-31(7/9)(\ln \lambda)F^{-i}}\right) \geq 1 - \sum_{i=0}^{R^*-1} 2\lambda^{-(217/9)F^{-i}}$ by the Weierstrass product inequality. Since the last sum equals $2\sum_{i=1}^{R^*} e^{-31F^i} \leq 2\sum_{i=1}^{R^*} e^{-31F \cdot i} \leq 2\frac{e^{-F}}{1-e^{-F}} \leq 4e^{-32}$, the probability is altogether at least $1 - 4e^{-32}$. Moreover, the probability of not flipping any zero-bits in at least one offspring, resulting in not increasing fitness distance, is for rate $r \leq (7/9)\ln \lambda$ at least

$$1 - \left(1 - \frac{1}{2}\left(1 - \frac{r}{Fn}\right)^n\right)^{\lambda/3} \geq 1 - \left(1 - \Omega(e^{-r})\right)^{\lambda/3} \geq 1 - e^{-\Omega(\lambda^{2/9})}$$

since the probability of using the smaller rate is 1/2. By a union bound over R^* iterations, the probability of decreasing the initial rate to at most F without losing fitness is at least $1 - 4e^{-32} - R^* e^{-\Omega(\lambda^{2/9})} \geq 5/6$ for sufficiently large n, using $\lambda = \omega(1)$. We assume this event to happen.

We are now in the position to apply the first item of Lemma 3.12, which yields that the drift at fitness distance $k \leq 3n/\lambda$ is at least $\gamma\lambda k/n$, where $\gamma > 0$ is a constant. We are interested in the expected time to reduce the fitness distance to 0. To this end, we use multiplicative drift analysis with starting point $2n/\lambda$ and drift factor λ/n to obtain a bound of $O(\frac{\ln(n/\lambda)n}{\lambda})$. By Markov's inequality, the number of generations is at most $t := c' \ln(n/\lambda)(n/\lambda)$ with probability at least 5/6, choosing c' as a sufficiently large constant. We assume this to happen.

We now apply the second item of Lemma 3.12 to estimate the total loss (i. e., increase of fitness distance) incurred during the drift phase of length t. By linearity of expectation, the expected value of this loss is $o(t(\max_{s=0,\dots,t} k_s)/n) = o(n/\lambda)$. By Markov's inequality, the loss is at most n/λ with probability at least 5/6. If this happens, the maximum OneMax-value during the phase is at most $3n/\lambda$ as required. Hence, assuming both a total loss of at most n/λ and that the drift phase lasts at most t generations, the OneMax-value is reduced to at most 0 within t generations. By a union bound, the success probability is at least $1 - 1/6 - 1/6 - 1/6 = 1/2$. □

3.3 Proof of Theorem 3.1

We can now put everything together to prove our main result.

Proof. Starting with arbitrary initialization, Theorem 3.9 along with a Markov bound yield that with probability $\Omega(1)$ after $t = O(n/\log\lambda)$ iterations a search point is reached such that $k_t \leq 2n/\lambda$ and $r_t < 0.6(\ln\lambda)$. Assuming this to happen, the assumptions of Theorem 3.13 are satisfied. Hence, after another $O((n\log n)/\lambda)$ iterations the optimum is found with probability at least 1/2. Altogether, with probability $\Omega(1)$ the optimum is found from an arbitrary initial OneMax-value and rate within $T^* = O(n/\log\lambda + (n\log n)/\lambda)$ iterations. The claimed expected time now follows by a standard restart argument, more precisely by observing that after expected $O(1)$ repetitions of a phase of length T^* the optimum is found. □

CONCLUSIONS

We have analyzed the self-adaptive (1,λ) EA using a very simple scheme for mutating the mutation rate and proved that is achieves the expected runtime $O(n\lambda/\log\lambda + n\log n)$ on OneMax, which is optimal for all λ-parallel mutation-based unbiased black-box algorithms. Hence, we have identified a simple and natural example where self-adaptation of strategy parameters in discrete EAs can lead to provably optimal runtimes that beat all known static parameter settings. Moreover, a relatively complicated and partly unintuitive self-adjusting scheme for the mutation rate proposed in [11] can be replaced by our simple endogenous scheme.

The analysis of the (1,λ) EA has revealed a highly non-trivial stochastic process where drift happens in two dimensions, namely regarding fitness distance and mutation rate. We have advanced the techniques for the analysis of such two-dimensional drift processes, including a useful lemma about occupation probabilities. Altogether, we are optimistic that our research helps pave the ground for further analyses of self-adaptive EAs.

ACKNOWLEDGMENTS

The authors thank Christian Gießen for useful discussions on this topic. This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences. This publication is based upon work from COST Action CA15140, supported by COST.

REFERENCES

[1] Thomas Bäck. 1992. Self-adaptation in genetic algorithms. In *Proc. of ECAL '92*. MIT Press, 263–271.
[2] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. 2014. Unbiased black-box complexity of parallel search. In *Proc. of PPSN '14*. Springer, 892–901.
[3] Sergey N. Bernstein. 1924. On a modification of Chebyshev's inequality and of the error formula of Laplace. *Ann. Sci. Inst. Sav. Ukraine, Sect. Math. 1* 4 (1924), 38–49.
[4] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. 2010. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proc. of PPSN '10*. Springer, 1–10.
[5] Duc-Cuong Dang and Per Kristian Lehre. 2016. Self-adaptation of mutation rates in non-elitist populations. In *Proc. of PPSN '16*. Springer, 803–813.
[6] Benjamin Doerr. 2011. Analyzing Randomized Search Heuristics: Tools from Probability Theory. In *Theory of Randomized Search Heuristics*, Anne Auger and Benjamin Doerr (Eds.). World Scientific, 1–20.
[7] Benjamin Doerr and Carola Doerr. 2015. Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings. In *Proc. of GECCO '15*. ACM, 1335–1342.
[8] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2013. Lessons from the black-box: fast crossover-based genetic algorithms. In *Proc. of GECCO '13*. ACM, 781–788.
[9] Benjamin Doerr, Carola Doerr, and Jing Yang. 2016. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proc. of PPSN '16*. Springer, 824–834.
[10] Benjamin Doerr, Carola Doerr, and Jing Yang. 2016. Optimal parameter choices via precise black-box analysis. In *Proc. of GECCO '16*. ACM, 1123–1130.
[11] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. 2017. The (1+λ) evolutionary algorithm with self-adjusting mutation rate. In *Proc. of GECCO '17*. ACM, 777–784. Full version available at http://arxiv.org/abs/1704.02191.
[12] Bruce Hajek. 1982. Hitting-time and occupation-time bounds implied by drift analysis with applications. *Advances in Applied Probability* 13 (1982), 502–525.
[13] Jens Jägersküpper. 2011. Combining Markov-chain analysis and drift analysis – the (1+1) evolutionary algorithm on linear functions reloaded. *Algorithmica* 59 (2011), 409–424.
[14] Thomas Jansen. 2013. *Analyzing Evolutionary Algorithms - The Computer Science Perspective*. Springer.
[15] Thomas Jansen and Ingo Wegener. 2006. On the analysis of a dynamic evolutionary algorithm. *Journal of Discrete Algorithms* 4 (2006), 181–199.
[16] Daniel Johannsen. 2010. *Random combinatorial structures and randomized search heuristics*. Ph.D. Dissertation. Saarland University.
[17] Giorgos Karafotias, Mark Hoogendoorn, and A.E. Eiben. 2015. Parameter control in evolutionary algorithms: trends and challenges. *IEEE Transactions on Evolutionary Computation* 19 (2015), 167–187.
[18] Timo Kötzing, Andrei Lissovoi, and Carsten Witt. 2015. (1+1) EA on generalized dynamic OneMax. In *Proc. of FOGA '15*. ACM, 40–51.
[19] Jörg Lässig and Dirk Sudholt. 2011. Adaptive population models for offspring populations and parallel evolutionary algorithms. In *Proc. of FOGA '11*. ACM, 181–192.
[20] Boris Mitavskiy, Jonathan E. Rowe, and Chris Cannings. 2009. 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 (2009), 243–284.
[21] Jonathan E. Rowe and Dirk Sudholt. 2014. The choice of the offspring population size in the (1, λ) evolutionary algorithm. *Theoretical Computer Science* 545 (2014), 20–38.