Optimal Static and Self-Adjusting Parameter Choices for the (1 + (λ , λ)) Genetic Algorithm

Benjamin Doerr, Carola Doerr

Algorithmica manuscript No.
(will be inserted by the editor)

Optimal Static and Self-Adjusting Parameter Choices for the $(1 + (\lambda, \lambda))$ Genetic Algorithm

Benjamin Doerr · Carola Doerr

Received: date / Accepted: date

Abstract The $(1+(\lambda,\lambda))$ genetic algorithm (GA) proposed in [Doerr, Doerr, and Ebel. From black-box complexity to designing new genetic algorithms. Theoretical Computer Science (2015)] is one of the few examples for which a super-constant speed-up of the expected optimization time through the use of crossover could be rigorously demonstrated. It was proven that the expected optimization time of this algorithm on OneMax is $O(\max\{n\log(n)/\lambda, \lambda n\})$ for any offspring population size $\lambda \in \{1,\dots,n\}$ (and the other parameters suitably dependent on λ) and it was shown experimentally that a self-adjusting choice of λ leads to a better, most likely linear, runtime.

In this work, we study more precisely how the optimization time depends on the parameter choices, leading to the following results on how to optimally choose the population size, the mutation probability, and the crossover bias both in a static and a dynamic fashion.

For the mutation probability and the crossover bias depending on λ as in [DDE15], we improve the previous runtime bound to $O(\max\{n\log(n)/\lambda, n\lambda\log\log(\lambda)/\log(\lambda)\})$. This expression is minimized by a value of λ slightly larger than what the previous result suggested and gives an expected optimization time of $O\left(n\sqrt{\log(n)\log\log\log(n)/\log\log(n)}\right)$.

We show that no static choice in the three-dimensional parameter space of offspring population, mutation probability, and crossover bias gives an asymptotically better runtime.

Results presented in this work are based on [12–14].

B. Doerr
École Polytechnique, LIX - UMR 7161, 1 rue Honoré d'Estienne d'Orves, Bâtiment Alan Turing, CS35003, 91120 Palaiseau, France

C. Doerr
Sorbonne Universités, UPMC Univ Paris 06, CNRS, LIP6 UMR 7606, 4 place Jussieu, 75005 Paris, France.

We also prove that the self-adjusting parameter choice suggested in [DDE15] outperforms all static choices and yields the conjectured linear expected runtime. This is asymptotically optimal among all possible parameter choices.

Keywords Theory of Randomized Search Heuristics · Runtime Analysis · Genetic Algorithms · Parameter Choice · Parameter Control

1 Introduction

The role of crossover in evolutionary computation is still not very well understood. On the one hand, it is used intensively and successfully in practice. On the other hand, there is only mildly convincing rigorous theoretical or experimental support for the usefulness of crossover. For example, attempts to experimentally support the building block hypothesis in [50] failed. On the theory side, there is a long series of works [6, 7, 22, 25, 28, 34, 42, 56, 57] presenting examples where crossover gives an improvement. A closer look at the details, however, often reveals that these work regard highly artificial problems or need uncommon parameter settings or substantial additional mechanisms to make crossover really work. Hence despite being positive examples, the overall impression that one might get could rather be that crossover is not that easily employed, or, at least, that our rigorous understanding of its working principles is not yet satisfactory.

In this work, we build on the latest algorithm where crossover was proven to be useful, the $(1+(\lambda,\lambda))$ genetic algorithm (GA) proposed in [15] (see [16] for the journal version). Unlike most previous works, here crossover has a superconstant speed-up even for very simple functions like the OneMax function $\textsc{Om} : \{0,1\}^n \to \{0,1,\dots,n\}, x \mapsto \sum_{i=1}^{n} x_i$. Experiments show that the algorithm performs well also on linear functions and royal road functions [16], on maximum satisfiability instances [36] (in fact, the latter work shows that the $(1+(\lambda,\lambda))$ GA outperforms hill climbers for several problems, while for MaxSat it also outperforms the Linkage Tree Genetic Algorithm) and on other combinatorial optimization problems [49].

The $(1+(\lambda,\lambda))$ GA not only shows an improved performance for various problems, it also uses crossover in a novel way. Instead of trying to combine particularly fit solution parts, it uses a biased uniform crossover with a parent individual as *repair mechanism*. With this repair mechanism, an increased mutation rate can be employed, leading to a faster exploration of the search space.

This way of using crossover, not seen before in discrete evolutionary optimization, motivates the quest for a deeper understanding of the $(1+(\lambda,\lambda))$ GA, its working principles, and the influence of static and dynamic parameter choices on the expected runtime. In the remainder of this introduction, we briefly recall the main ideas of the $(1+(\lambda,\lambda))$ GA and its parameters in Section 1.1. Our results are summarized in the subsequent Sections 1.2 (stronger runtime guarantee for the parameters suggested in [16]), 1.3 (lower bound for

the full 3-dimensional parameter space), and 1.4 (upper and lower bounds for a self-adjusting parameter choice). The main part of the paper follows a similar outline, with the $(1 + (\lambda, \lambda))$ GA made precise in Section 2, some technical tools prepared in Section 3, and then the results described in Section 1.2 to 1.4 being proven in Sections 4 to 6, respectively.

1.1 The $(1 + (\lambda, \lambda))$ Genetic Algorithm

The $(1+(\lambda, \lambda))$ GA works with a parent population of size one. This population $\{x\}$ is initialized with a search point chosen from $\{0, 1\}^n$ uniformly at random. The $(1 + (\lambda, \lambda))$ GA then proceeds in iterations, each consisting of a mutation phase, a crossover phase, and a final elitist selection step determining the new parent population.

In the *mutation phase*, a step size ℓ is chosen at random from the binomial distribution $\mathcal{B}(n, p)$, where p denotes the mutation probability. Then independently λ offspring are sampled by flipping exactly ℓ random bits in x. In an intermediate selection step, one best mutation offspring x' is selected as mutation winner. In the *crossover phase*, again λ offspring are created; this time via a biased uniform crossover between x and x', taking each entry of x' with probability c only and taking the entry of x otherwise. Again, an intermediate selection chooses one of the best crossover offspring y as crossover winner. In the final *selection step*, this y replaces x if its *fitness* is at least as large as the fitness of x; i.e., if and only if $f(y) \geq f(x)$ holds.

The three parameters determining the $(1+(\lambda, \lambda))$ GA are thus the offspring population size λ, the mutation probability p, and the crossover bias c. Using intuitive considerations, in [16] it was suggested to generally use $p = \lambda/n$ and $c = 1/\lambda$.

1.2 An Improved Upper Bound

Our first result is a refined runtime analysis for the problem analyzed in [16], that is, for all $\lambda \leq n$ we regard the expected runtime (expected number of fitness evaluations until an optimal solution is found) of the $(1 + (\lambda, \lambda))$ GA on OneMax for $p = \lambda/n$ and $c = 1/\lambda$. While in [16] an upper bound of

$$O\left(\max\left\{\frac{n\log(n)}{\lambda}, \lambda n\right\}\right)$$

was shown, we determine in this work the precise order of magnitude and show in Section 4 an upper bound of order

$$O\left(\max\left\{\frac{n\log(n)}{\lambda}, \frac{n\lambda\log\log(\lambda)}{\log(\lambda)}\right\}\right)$$

for all values of $\lambda \in \{1, \dots, n\}$.

While for the previous bound from [16] a parameter setting of $\lambda = \Theta(\sqrt{\log n})$ is the choice giving the minimal expected runtime, namely $O(n\sqrt{\log n})$, our new result suggests that a slightly larger value of

$$\lambda = \Theta(\sqrt{\log(n)\log\log(n)/\log\log\log(n)})$$

is superior, which gives an expected optimization time of

$$O(n\sqrt{\log(n)\log\log\log(n)/\log\log(n)}).$$

Our (more general) lower bound of Section 5 will show that both this runtime and this choice for λ are asymptotically optimal.

We further prove a strong concentration result for the runtime, showing that deviations above the expectation are unlikely: For all $\delta > 0$, the probability that the actual runtime exceeds its expected value by a factor of $(1+\delta)$ is at most $O((n/\lambda^2)^{-\delta})$.

The proofs of these results also give some additional insights into the working principles of the crossover operator and, more generally, the $(1+(\lambda,\lambda))$ GA. The improved runtime guarantee is based on the observation that when generating λ offspring in parallel, some have a fitness that is significantly better than the expected fitness of a single offspring created by the biased crossover. We exploit this to show that sufficiently often we gain sufficiently many fitness levels in one iteration, where we use the common convention to partition the search space $\{0,1\}^n$ into the $n+1$ fitness levels $L_0, L_1, \dots, L_n$ such that for all $i \in [0..n]$ the *fitness level* L_i contains exactly those strings x with $\textsc{Om}(x) = i$; i.e., those strings that have exactly i entries equal to 1. Interestingly, the good runtimes shown for the $(1 + (\lambda,\lambda))$ GA only stem from better-than-expected individuals in the crossover phase, but not from such individuals in the mutation phase.

With the few runtime results on crossover-based evolutionary algorithms (EAs) and, also, still the majority of the runtime results for EAs in general being for algorithms having trivial population sizes, we feel that our work also advances the state of the art in terms of analytical methods. Our argument that one out of λ offspring can have a significantly better fitness than the expected fitness of one offspring resembles a similar one previously made by Jansen, De Jong, and Wegener [41], who used multiple fitness level gains to prove that for the $(1 + \lambda)$ EA optimizing OneMax a linear speed-up (compared to $\lambda = 1$) exists if and only if $\lambda = O(\log(n)\log\log(n)/\log\log\log(n))$. An extension of this analysis in [27] gives a tight runtime bound—of $\Theta(n\lambda\log\log(\lambda)/\log(\lambda))$—also for the regime where λ is above the cut-off point $\Theta(\log(n)\log\log(n)/\log\log\log(n))$. This analysis also exploits gains of multiple fitness levels by the best of several independent offspring. Note that both results are different from ours in two respects, namely in that they do not show a positive influence of λ on the expected optimization time (number of fitness evaluations), but only on the number of iterations, and in that the better-than-expected offspring are generated by mutation, and not by crossover. A difference in terms of proof methods is that

the random experiment producing the new generation for the $(1 + (\lambda, \lambda))$ GA has stochastic dependencies that are not present in the $(1 + \lambda)$ EA, thus requiring a number of different arguments. For example, we cannot simply re-use classic results on balls-into-bins experiments, but have to re-prove them taking into account the dependencies present in our random experiment. To show the strong concentration result, we prove a Chernoff-type large deviation bound for sums of geometric random variables having harmonically decreasing expectations (Lemma 4). Since such random variables occur frequently in the analysis of randomized search heuristics, we are optimistic that this tool will find applications in other works.

1.3 Optimal Static Parameter Choices—Tight Bounds for A Multi-Dimensional Parameter Space

The refined analysis above (together with the matching lower bound given in [14]) determines the optimal choice of the population size λ when $p = \lambda/n$ and $c = 1/\lambda$ are chosen according to the intuitive arguments given in [16]. It thus leaves open the possibility that completely different parameter choices give an even better performance.

For this reason, in Section 5 we rigorously prove a lower bound valid for the whole three-dimensional parameter space. Our lower bound coincides with the runtime proven in Section 4. Consequently, the intuitive dependencies of the parameters p and c suggested in [16] are indeed optimal. Our result and in particular the partial results that lead to it give a clearer picture on how to choose the parameters in the $(1 + (\lambda, \lambda))$ GA also for optimization problems beyond the OneMax function class. We discuss these non-rigorous insights in Section 5.3.

From the methodological standpoint, our analysis is one of very few theoretical works that analyze evolutionary algorithms involving more than one parameter. We observe that the parameters do not have an independent influence on the runtime, but that they interact in a difficult to foresee manner. A similar observation was made in [35], where it is proven for the $(1 + \lambda)$ EA that the mutation probability has a decisive influence on the performance when the population size λ is asymptotically smaller than the cut-off point $\log(n) \log\log(n) / \log\log\log(n)$, whereas it has almost no influence when $\lambda = \omega(\log(n) \log\log(n) / \log\log\log(n))$. Such non-separable parameter influences, naturally, make the analysis of a multi-dimensional parameter space more difficult.

A second difficulty we had to overcome is that, while only a few parameter configurations yield the asymptotically optimal expected runtime, quite a large set of parameter combinations (including some that are far from the optimal ones) still lead to an expected runtime that is very close to the optimal one (see the remark at the end of Section 5.1). While this is good from the application point of view (missing the optimal parameters is not necessarily devastating), from the viewpoint of proving our results it means that there is not much room

for non-sharp estimates. In overcoming these difficulties, we are optimistic that our work helps future analyses of multi-dimensional parameter spaces.

1.4 Self-Adjusting Parameter Choices

All results described above assume *static* choices of the parameters λ, p, and c; that is, these values remain unchanged for the whole run of the algorithm. It had been proven already in [16] that a better expected runtime, namely a linear one, can be achieved if we allow the parameters to depend on the current state of the optimization process. More precisely, it was shown that for $p = \lambda/n$, $c = 1/\lambda$, and

$$\lambda = \sqrt{n/(n - f(x))} \tag{1}$$

with $f(x)$ denoting the fitness of the current search point, the resulting expected optimization time of the $(1 + (\lambda, \lambda))$ GA on OneMax is $\Theta(n)$. While this shows that the best parameter setting changes during the course of the optimization process, assuming that an algorithm user is able to guess a relation like (1) is not very realistic.

However, in [16] also a second, easier to use idea has been suggested: a discrete version of the well-known one-fifth success rule (see the discussion in Section 6.2) has experimentally been shown capable of tracking the theoretically optimal parameter values in an automated way, thus obtaining an expected runtime that seems to be linear. We make these observations rigorous in Section 6 and prove (Theorem 9) that the self-adjusting $(1+(\lambda, \lambda))$ GA suggested in [16] has a linear expected optimization time on OneMax. By the results of Section 5 this is faster than what any static parameter choice can achieve. We also prove that this linear expected runtime is optimal among all possible dynamic parameter settings (Theorem 11).

2 The $(1 + (\lambda, \lambda))$ GA and Its Parameters

The $(1 + (\lambda, \lambda))$ GA, which was originally proposed in [15] (see [16] for the full journal version), is an evolutionary algorithm using crossover. Its pseudocode is given in Algorithm 1. The algorithm is initialized by forming the parent population with a solution candidate x that is drawn uniformly at random from $\{0, 1\}^n$. The $(1 + (\lambda, \lambda))$ GA proceeds in iterations consisting of a mutation, a crossover, and a selection phase. In an important contrast to many other genetic algorithms, the mutation phase *precedes* the crossover phase. This allows to use crossover as a repair mechanism, as we shall discuss in more detail below.

In the *mutation phase*, we create λ offspring from the parent x by applying to it the mutation operator $\mathrm{mut}_\ell(\cdot)$, which flips in x the entries in ℓ positions that are selected from $[n] := \{1, 2, \dots, n\}$ uniformly at random. In other words, $\mathrm{mut}_\ell(x)$ is a bit-string in which for ℓ random positions i the entry $x_i \in \{0, 1\}$ is

Algorithm 1: The $(1 + (\lambda, \lambda))$ GA, maximizing a given function $f : \{0,1\}^n \to \mathbb{R}$, with offspring population size λ, mutation probability p, and crossover bias c. The mutation operator mut_ℓ generates an offspring from one parent by flipping exactly ℓ random bits (without replacement). The crossover operator cross_c performs a biased uniform crossover, taking bits independently with probability c from the second argument.

Initialization: Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.);
Optimization: **for** $t = 1, 2, 3, \ldots$ **do**
  Mutation phase:
    Sample ℓ from $\mathcal{B}(n, p)$;
    for $i = 1, \ldots, \lambda$ **do** $x^{(i)} \leftarrow \text{mut}_\ell(x)$;
    Choose $x' \in \{x^{(1)}, \ldots, x^{(\lambda)}\}$ with $f(x') = \max\{f(x^{(1)}), \ldots, f(x^{(\lambda)})\}$ u.a.r.;
  Crossover phase:
    for $i = 1, \ldots, \lambda$ **do** $y^{(i)} \leftarrow \text{cross}_c(x, x')$;
    Choose $y \in \{y^{(1)}, \ldots, y^{(\lambda)}\}$ with $f(y) = \max\{f(y^{(1)}), \ldots, f(y^{(\lambda)})\}$ u.a.r.;
  Selection step: **if** $f(y) \geq f(x)$ **then** $x \leftarrow y$;

replaced by $1 - x_i$. The *step size* ℓ is chosen randomly according to a binomial distribution $\mathcal{B}(n, p)$ with n trials and success probability p. To ensure that all mutants have the same distance from the parent x, and thus to not bias the selection by different distances from the parent, the same ℓ is used for all λ offspring. The fitness of the λ offspring is computed and the best one of them, x', is selected to take part in the crossover phase. If there are several offspring having maximal fitness, we pick one of them uniformly at random (u.a.r.).

When the parent x is already close to an optimal solution, the offspring created in the mutation phase are typically all of much worse fitness than x. However, they may still have discovered some parts of an optimal solution that is not yet reflected in x. In order to preserve these parts while at the same time not destroying the good parts of x, the $(1+(\lambda, \lambda))$ GA creates in the *crossover phase* λ offspring from x and x'. Each one of these offspring is sampled from a uniform crossover with bias c to take an entry from x'; that is, each offspring $y^{(i)} := \text{cross}_c(x, x')$ is created by setting $y_j^{(i)} := x'_j$ with probability c and taking $y_j^{(i)} := x_j$ otherwise, independently for each position $j \in [n]$. Again we evaluate the fitness of the λ crossover offspring and select the best one of them, which we denote by y. If there are several offspring of maximal fitness, we break ties randomly.[1]

Finally, in the *selection step* the parent x is replaced by the winner of the intermediate selection of the crossover phase y if and only if the fitness of y is at least as good as the one of x.

[1] In [15, Section 4.4] and [16] a slightly different selection rule was suggested for the crossover phase, namely excluding from the crossover selection individuals identical with x. This can speed-up traversing large plateaus of equal fitness. For the OneMax function, naturally, there is no difference, so we present the algorithm in the simpler fashion given above.

As common in the runtime analysis community, we do not specify a termination criterion. The simple reason is that we study as a theoretical performance measure the expected number of function evaluations that the $(1+(\lambda,\lambda))$ GA performs until it evaluates for the first time a search point of maximal fitness (the so-called *optimization time* or *runtime*). This is the common measure in runtime analysis, cf. Section 3.1 for a discussion. Note that for algorithms performing more than one fitness evaluation per iteration, such as the $(1+(\lambda,\lambda))$ GA, the expected runtime can be much different from the expected number of *iterations* (generations). Of course, for an application to a real problem a termination criterion has to be specified for the $(1+(\lambda,\lambda))$ GA.

2.1 Parameter Choices

The $(1+(\lambda,\lambda))$ GA comes with a set of parameters, namely the mutation probability p, the crossover bias c, and the offspring population size λ. If $\ell \sim \mathcal{B}(n,p)$, then $\text{cross}_c(x, \text{mut}_\ell(x))$ has the distribution of an individual created from x via standard bit mutation with mutation rate pc. Since $1/n$ is an often preferred choice for the mutation rate, it was suggested in [16] to choose p and c in a way that $pc = 1/n$. Note, however, that due to the two intermediate selection steps, the final offspring y of one iteration of the $(1+(\lambda,\lambda))$ GA has a very different distribution than standard bit mutation with rate pc. For example, as we will see in Section 4, for all x, apart from those stemming from $o(n)$ fitness levels, the final offspring y gains a super-constant number of fitness levels over x.

We parametrize $p = k/n$ so that k denotes the average number of bits flipped by an application of the mutation operator. With this setting, the above suggestion translates to choosing $c = 1/k$. For these settings, a first runtime analysis for the OneMax test function in [16] gave an upper bound for the expected runtime of $O((\frac{1}{k}+\frac{1}{\lambda})n\log n + (k+\lambda)n)$. From this, the suggestion to take $k = \lambda$ was derived, reducing the parameter space to the single parameter λ. Since only an upper bound for the expected runtime was used to obtain this suggestion, again this is an intuitive argument, but not a rigorous one.

For the parameter settings $p = \lambda/n$, $c = 1/\lambda$, and arbitrary λ we shall perform a more precise runtime analysis in Section 4 showing a order of magnitude for the expected runtime of

$$O\left(\max\left\{\frac{n\log(n)}{\lambda}, \frac{n\lambda\log\log(\lambda)}{\log(\lambda)}\right\}\right), \tag{2}$$

which is minimized by the parameter choice

$$\lambda = \Theta\left(\sqrt{\log(n)\log\log(n)/\log\log\log(n)}\right).$$

In Section 5 we shall prove that the upper bound (2) is tight and that, furthermore, also all other choices of mutation probability, crossover bias, and offspring population size lead to this or a worse expected runtime.

For dynamic parameter choices, i.e., for parameter choices that can change during the optimization process, we recall that [16] showed that a $\Theta(n)$ expected runtime can be obtained by choosing the parameters using a clever functional dependence of the objective value. It was also shown experimentally that a self-adjusting parameter setting imitating a one-fifth success rule yields very similar performance. In Section 6 we shall formally prove that, indeed, the $(1 + (\lambda, \lambda))$ GA together with this update scheme has a linear expected optimization time on OneMax.

3 Notation and Technical Tools

In this section, besides fixing some very elementary notation, we collect the main technical tools we shall use. Mostly, these are large deviations bounds of various types. For the convenience of the reader, we first state the known ones. We then prove a tail bound for sums of geometric random variables with expectations bounded from above by the reciprocals of the first positive integers. We finally recall the well-known additive drift theorem as well as a multiplicative drift theorem for computing lower bounds.

3.1 Runtime Analysis

Runtime analysis is one of the most successful theoretical tools to understand the performance of evolutionary algorithms. The *runtime* or *optimization time* of an algorithm (e.g., our $(1 + (\lambda, \lambda))$ GA) on a problem instance (e.g., the OneMax function) is the number of fitness evaluations that are performed until an optimal solution is evaluated for the first time. If the algorithm is randomized (like our $(1 + (\lambda, \lambda))$ GA), this is a random variable T, and we usually make statements on the expected value $E[T]$ or give bounds that hold with some high probability, e.g., $1 - 1/n$. When regarding a problem with more than one instance (e.g., traveling salesman instance on n cities), we take a worst-case view. This is, we regard the maximum expected runtime over all instances, or we make statements like that the runtime satisfies a certain bound for all instances.

In this work, the optimization problem we regard is the classic OneMax test problem consisting of the single instance $\textsc{Om} : \{0,1\}^n \to \{0,1,\dots,n\}; x \mapsto \sum_{i=1}^{n} x_i$, that is, the optimization goal is to maximize the number of ones in a bit-string. Despite the simplicity of the OneMax problem, analyzing randomized search heuristics on this function has spurred much of the progress in the theory of evolutionary computation in the last 20 years, as is documented, e.g., in the recent textbook [40].

Of course, when regarding the performance on a single test instance, we should ensure that the algorithm does not exploit the fact that there is only one instance. A counter-example would be the algorithm that simply evaluates and outputs $x^* = (1,\dots,1)$, giving a perfect runtime of 1. One way of ensuring

this is that we restrict ourselves to unbiased algorithms (see [48]) which treat bit-positions and bit-values in a symmetric fashion. Consequently, an unbiased algorithm for the OneMax problem has the same performance on all problems with isomorphic fitness landscape, in particular, on all (generalized) OneMax functions $\textsc{Om}_z : \{0,1\}^n \to \{0,1,\dots,n\}; x \mapsto \mathrm{eq}(x,z)$ for $z \in \{0,1\}^n$, where $\mathrm{eq}(x,z)$ denotes the number of bit-positions in which x and z agree. It is easy to see that the $(1+(\lambda,\lambda))$ GA is unbiased (for all parameter settings).

3.2 Notation and Two Elementary Facts

We write $[a..b]$ to denote the set $\{z \in \mathbb{Z} \mid a \leq z \leq b\}$ of integers between a and b. We write $\log(n)$ to denote the binary logarithm of n and $\ln(n)$ to denote the natural logarithm of n. However, to avoid unnecessary case distinctions when taking iterated logarithms, we define $\log(n) := 1$ for all $n \leq 2$ and $\ln(n) := 1$ for all $n \leq e$. For the readers' convenience, we now collect some tools from probability theory which we will use regularly.

We occasionally need the expected value of a binomially distributed random variable $X \sim \mathcal{B}(n,p)$ conditional on that the variable has at least a certain value k. An intuitive (but wrong) solution to this question is that this $E[X|X \geq k]$ should be around $k + p(n-k)$, because we know already that at least k of the n independent trials are successes and the remaining $(n-k)$ trials still have their independent success probability of p. While this argument is wrong, an upper bound of this type can be shown by elementary means. Since we have not seen this made explicit in the EA literature, we shall also give the short proof.

Lemma 1 *Let X be a random variable with binomial distribution with parameters n and $p \in [0,1]$. Let $k \in [0..n]$. Then*

$$E[X \mid X \geq k] \leq k + (n-k)p \leq k + E[X].$$

Proof Let $X_1, \dots, X_n$ be independent binary random variables with $\Pr[X_i = 1] = p$ for all $i \in [n]$. Then $X = \sum_{i=1}^n X_i$ has a binomial distribution with parameters n and p. Conditioning on $X \geq k$, let $\ell := \min\{i \in [n] \mid \sum_{j=1}^i X_j = k\}$. Then $E[X \mid X \geq k] = \sum_{i=1}^n \Pr[\ell = i \mid X \geq k] E[X \mid \ell = i]$. Note that $\ell \geq k$ by definition. Note also that $\mathrm{E}[X \mid \ell = i] = k + \sum_{j=i+1}^n X_j$ with unconditioned X_j. In particular, $E[X \mid \ell = i] = k + (n-i)p$. Consequently, $E[X \mid X \geq k] = \sum_{i=1}^n \Pr[\ell = i \mid X \geq k] E[X \mid \ell = i] \leq \sum_{i=k}^n \Pr[\ell = i \mid X \geq k](k + (n-k)p) = k + (n-k)p$. □

Also, we shall use the following well-known fact, for which a short proof can be found, for example, in [20, Lemma 1].

Lemma 2 *Let X be a non-negative integral random variable. Then $E[X] = \sum_{i=1}^\infty \Pr[X \geq i]$.*

3.3 Chernoff Bounds

The following *large deviation bounds* are well known and can be found, e.g., in [11]. They are often called *Chernoff bounds* despite the fact that it is well known that many of them have been discovered by Bernstein, Hoeffding, and others.

Theorem 1 (classic Chernoff bounds) *Let $X_1, \dots, X_n$ be independent random variables taking values in $[0, 1]$. Let $X = \sum_{i=1}^{n} X_i$.*

(a) Let $\delta \geq 0$. Then $\Pr[X \geq (1+\delta)E[X]] \leq \left(\frac{e^\delta}{(1+\delta)^{1+\delta}}\right)^{E[X]}$.
(b) Let $\delta \in [0, 1]$. Then $\Pr[X \geq (1+\delta)E[X]] \leq \exp(-\delta^2 E[X]/3)$.
(c) Let $d \geq 6E[X]$. Then $\Pr[X \geq d] \leq 2^{-d}$.
(d) Let $\delta \in [0, 1]$. Then $\Pr[X \leq (1-\delta)E[X]] \leq \exp(-\delta^2 E[X]/2)$.
(e) Let $X_1, \dots, X_n$ be independent random variables each taking values in some interval of length at most one. Let $X = \sum_{i=1}^{n} X_i$. Let $\lambda \geq 0$. Then $\Pr[X \leq E[X] - \lambda] \leq \exp(-2\lambda^2/n)$ and $\Pr[X \geq E[X] + \lambda] \leq \exp(-2\lambda^2/n)$.

Binary random variables $X_1, \dots, X_n$ are called *negatively correlated*, if for all $I \subseteq [n]$ we have $\Pr[\forall i \in I : X_i = 0] \leq \prod_{i \in I} \Pr[X_i = 0]$ and $\Pr[\forall i \in I : X_i = 1] \leq \prod_{i \in I} \Pr[X_i = 1]$.

Theorem 2 (Chernoff bound, negative correlation) *Let $X_1, \dots, X_n$ be negatively correlated binary random variables. Let $a_1, \dots, a_n \in [0, 1]$ and $X = \sum_{i=1}^{n} a_i X_i$. Then X satisfies the Chernoff bounds given in Theorem 1 (b) and (d).*

Chernoff bounds also hold for *hypergeometric* distributions. Let A be any set of n elements. Let B be a subset of A having m elements. If Y is a random subset of A of N elements, chosen uniformly at random from all N-element subsets of A, then $X := |Y \cap B|$ has a hypergeometric distribution with parameters (n, N, m).

Theorem 3 (Chernoff bounds for hypergeometric distributions) *If X has a hypergeometric distribution with parameters (n, N, m), then $E[X] = Nm/n$ and X satisfies all Chernoff bounds given in Theorem 1.*

A different way of applying Chernoff bounds to random variables that are not fully independent is the following lemma proven in [11].

Lemma 3 (Chernoff bound, lower tail, moderate indepencence) *Let $X_1, \dots, X_n$ be arbitrary binary random variables. Let $X_1^*, \dots, X_n^*$ be binary random variables that are mutually independent and such that for all i, X_i^* is independent of $X_1, \dots, X_{i-1}$. Assume that for all i and all $x_1, \dots, x_{i-1} \in \{0, 1\}$,*

$$\Pr(X_i = 1 | X_1 = x_1, \dots, X_{i-1} = x_{i-1}) \geq \Pr(X_i^* = 1).$$

Then for all $k \geq 0$, we have

$$\Pr\left(\sum_{i=1}^{n} X_i < k\right) \leq \Pr\left(\sum_{i=1}^{n} X_i^* < k\right),$$

and the latter term can be bounded by Chernoff bounds for independent random variables.

3.4 A Chernoff Bound for Geometric Random Variables

To prove the concentration statement in Theorem 7, we need a tail bound for the upper tail of a sum of a sequence of independent geometric random variables having expectations that are upper-bounded by a multiple of the harmonic series. While generally Chernoff bounds for geometric random variables are less understood than for bounded random variables, Witt [62] proves such a bound. Witt's bound is sufficient for our purposes. For two reasons we prove the following alternative result below. (i) Our proof is a simple reduction to the well-understood coupon collector process, and thus much simpler than Witt's. (ii) At the same time, our proof gives a stronger bound (for our setting, Witt's bound on the failure probability is roughly the fourth root of ours). Since scenarios as treated here are quite common in runtime analysis (for example, they appear whenever the fitness level method is employed in a situation where the probability of a progress is inversely proportional to the fitness distance from the optimum), we feel that presenting our result is justified here.

We say that X has a geometric distribution with success probability p if for each positive integer k we have $\Pr[X = k] = (1-p)^{k-1}p$. For all $n \in \mathbb{N}$, let $H_n := \sum_{i=1}^{n}(1/i)$ denote the nth Harmonic number.

Lemma 4 *Let $X_1, \dots, X_n$ be independent geometric random variables with success probabilities p_i. Assume that there is a number $C \leq 1$ such that $p_i \geq Ci/n$ for all $i \in [n]$. Let $X = \sum_{i=1}^{n} X_i$. Then*

$$E[X] \leq (1/C)nH_n \leq (1/C)n(\ln(n) + 1)$$

and

$$\Pr[X \geq (1+\delta)(1/C)n\ln(n)] \leq n^{-\delta} \text{ for all } \delta > 0.$$

Proof For $i \in [n]$, let X_i' be a geometric random variable with success probability exactly $Ci/n =: p_i'$. Let the X_i' be independent. Then X_i' dominates X_i for all $i \in [n]$, and consequently, $X' := \sum_{i=1}^{n} X_i'$ dominates X. Recall that a random variable Y' dominates a random variable Y if for all $r \in \mathbb{R}$, $\Pr[Y \geq r] \leq \Pr[Y' \geq r]$. Note that this implies that $E[Y] \leq E[Y']$ and that all upper tail bounds for Y' immediately apply to Y. Consequently, we can conveniently focus on X' instead of X.

For the statement on the expectation, we recall that the expectation of a geometric random variable with success probability p is $1/p$. Consequently,

by linearity of expectation, we have $E[X'] = \sum_{i=1}^{n} E[X'_i] = \sum_{i=1}^{n}(1/p'_i) = (n/C)\sum_{i=1}^{n}(1/i) = (n/C)H_n$.

For the tail bound, consider the following coupon collector process. There are n different types of coupons. In each round (independently), with probability C we obtain a coupon having a type chosen uniformly at random and with probability $1 - C$ we obtain nothing. We are interested in the number T of rounds until we have each type of coupon at least once. For $i \in [n]$, let T_i denote the number of rounds needed to get a coupon of a type not yet in our possession given that we have already $n - i$ different types. In other words, T_i is the time we need to go from "i types missing" down to "$i-1$ types missing". We observe that T_i has the same distribution as X_i and that $T = \sum_{i=1}^{n} T_i$. Consequently, T and X are equally distributed.

The advantage of this reformulation is that it allows us a different view on $X' \sim T$: The probability that after t rounds of the coupon collector process we do not have a fixed type is exactly $(1 - C/n)^t$. Using a union bound, we see that the probability that after t rounds some coupon is missing, is at most $n(1 - C/n)^t$. For $t = (1 + \delta)(1/C)n\ln(n)$, this is at most $n(1 - C/n)^t \leq n\exp(-Ct/n) = n\exp(-(1 + \delta)\ln(n)) = n^{-\delta}$. □

Note that in the proof of Lemma 4, once we have defined the coupon collector process (but not before), we could have also used multiplicative drift. This would, however, not have given a better bound, nor a shorter proof.

3.5 Drift Analysis

Drift analysis comprises a couple of methods to use information about the expected progress (e.g., in terms of the fitness distance) to derive results about the time needed to achieve a goal (e.g., finding an optimal solution). We shall use several times the following *additive drift* theorem from [38] (see also Theorem 2.7 in [52]).

Theorem 4 (additive drift theorem, [38]) *Let $X_0, X_1, \ldots$ be a sequence of random variables taking values in a finite set $S \subseteq \mathbb{R}_{\geq 0}$. Let $T := \min\{t \geq 0 \mid X_t = 0\}$. Let $\delta > 0$.*

(i) If for all t, we have $E[X_t - X_{t+1} | X_t > 0] \geq \delta$, then $E[T|X_0] \leq X_0/\delta$.
(ii) If for all t, we have $E[X_t - X_{t+1} | X_t > 0] \leq \delta$, then $E[T|X_0] \geq X_0/\delta$.

In many situation, the progress $X_t - X_{t+1}$ is stronger when the process is far from the target, that is, when X_t is large. A particular, but seemingly very common special case is that the progress is indeed proportional to X_t. Such a situation is called *multiplicative drift*. Drift theorems giving upper bounds for the hitting time were given in [26] and [21]. Transforming upper bounds on a multiplicative progress into good lower bounds for hitting times requires additional assumptions. Witt gives the following very useful theorem (Theorem 2.2 in [61]). While not important for our purposes, we note that a version of this theorem not requiring the process to move only in one direction (that is, satisfying $X_t \geq X_{t+1}$ for any $t \geq 0$), was proven in [18].

Theorem 5 (multiplicative drift, lower bound, [61]) *Let $S \subset \mathbb{R}$ be a finite set of positive numbers with minimum 1. Let $X_0, X_1, \dots$ be a sequence of random variables over S such that $X_t \geq X_{t+1}$ for any $t \geq 0$. Let $s_{\min} > 0$. Let T be the random variable that gives the first point in time $t \geq 0$ for which $X_t \leq s_{\min}$. If there exist positive reals $\beta, \delta \leq 1$ such that, for all $s > s_{\min}$ and all $t \geq 0$ with $\Pr[X_t = s] > 0$,*

(1) $E[X_t - X_{t+1} | X_t = s] \leq \delta s$,
(2) $\Pr[X_t - X_{t+1} \geq \beta s | X_t = s] \leq \beta\delta / \ln(s)$,

then for all $S_0 \in S$ with $\Pr[X_0 = s_0] > 0$, we have

$$E[T | X_0 = s_0] \geq \frac{\ln(s_0) - \ln(s_{\min})}{\delta} \cdot \frac{1-\beta}{1+\beta}.$$

4 A Tight Upper Bound

As discussed in Section 2.1 the initial analysis of the $(1+(\lambda,\lambda))$ GA on OneMax presented in [16] suggested to use as mutation probability $p = \lambda/n$ and as crossover bias $c = 1/\lambda$. For these settings, the following upper bound for the expected runtime was proven (Theorem 4 in [16] for $\lambda = k$, note that the case $\lambda = 1$ excluded there is trivial for $\lambda = k$ since in this case, the $(1+(\lambda,\lambda))$ GA imitates the $(1+1)$ EA).

Theorem 6 ([16]) *Let $\lambda \in [n]$, possibly depending on n. The expected optimization time of the $(1+(\lambda,\lambda))$ GA with mutation probability $p = \lambda/n$ and crossover bias $c = 1/\lambda$ on* OneMax *is*

$$O\left(\max\left\{\frac{n \log n}{\lambda}, \lambda n\right\}\right).$$

In particular, for $\lambda = \Theta(\sqrt{\log n})$, the expected optimization time is of order at most $n\sqrt{\log n}$.

In this section we improve this bound mildly (but to the asymptotically correct order of magnitude, as we will see in Section 5) and add a tail bound showing that deviations above our guarantee on the expected runtime are extremely rare. We note without explicit proof that *the following result also holds for the natural modification of the algorithm in which a best individual among all mutation* and *crossover offspring competes with the parent individual in the main selection step*. This is immediately clear from our proofs, since they only use fitness level and drift arguments.

Theorem 7 *Let $\lambda \in [n]$. The expected optimization time of the $(1+(\lambda,\lambda))$ GA with mutation probability $p = \lambda/n$ and crossover bias $c = 1/\lambda$ on the* OneMax *test function is*

$$O\left(\max\left\{\frac{n \log(n)}{\lambda}, \frac{n\lambda \log\log(\lambda)}{\log(\lambda)}\right\}\right). \tag{3}$$

For all $\delta > 0$, the probability that the actual runtime exceeds a bound of this magnitude by a factor of more than $(1 + \delta)$ is at most $O((n/\lambda^2)^{-\delta})$.

The bound on the expected runtime is minimized by the parameter choice

$$\lambda = \Theta\left(\sqrt{\log(n)\log\log(n)/\log\log\log(n)}\right). \tag{4}$$

This yields an expected optimization time of

$$O\left(n\sqrt{\log(n)\log\log\log(n)/\log\log(n)}\right).$$

As mentioned above, we shall see in Section 5 that the bound in Theorem 7 is tight. While the proof in Section 5 provides a general lower bound for the whole three-dimensional parameter space, a much simpler proof for the tightness of (3) can be found in [14]. We do not repeat this proof here as is it is subsumed by the results of Section 5. However, we remind the reader that the tightness of (3) implies that the choice of λ in (4) is asymptotically optimal whenever p and c follow the dependency on λ suggested in [16].

In addition to showing that the $(1 + (\lambda, \lambda))$ GA is faster than what could be shown in [16], and providing the asymptotically optimal value for the offspring population size λ, our sharp bounds also give more insight into the working principles of this algorithm. In particular, we shall observe that in its crossover phase generating λ offspring in parallel often produces at least one offspring that is significantly better that the expected outcome of a crossover application. This allows to gain several (including regularly a super-constant number in the early time of the optimization process) fitness levels in one iteration. This advantage of larger offspring population sizes seems to have been rarely analyzed rigorously (with the analyses of the $(1+\lambda)$ EA in [27,41] being the only exceptions known to us). The more common use of larger offspring population sizes in the literature seems to be that an offspring population size of λ reduces the waiting time for a fitness level gain by approximately a factor of λ (given that this waiting time is large enough). This latter argument, naturally, does not reduce the (total) expected optimization time (number of fitness evaluations), but only the parallel one (number of iterations).

We now proceed with proving Theorem 7, that is, that the $(1 + (\lambda, \lambda))$ GA with standard parameter settings optimizes every OneMax function using $O\left(\max\left\{\frac{n\log(n)}{\lambda}, \frac{n\lambda\log\log(\lambda)}{\log(\lambda)}\right\}\right)$ fitness evaluations both in expectation and with probability $1 - n^{-c}$, where c is an arbitrary positive constant.

The proof of the previous upper bound (Theorem 6) was based on the fitness level method (first used in [58] in the proof of Theorem 1, more explicit in [59], see also [52]). In its classic version, this method pessimistically estimates the expected runtime via the sum of the times needed to leave each fitness level. It thus does not profit from the fact that a typical run of the algorithm might not visit every fitness level. By a more careful analysis of the mutation phase (Lemma 5) and the crossover phase (Lemma 6), we shall show that this indeed happens. For all values of λ, we obtain that when starting an iteration with a search point x having *fitness distance*

$d(x) := n - \textsc{Om}(x)$ at least $n \log\log(\lambda)/\log(\lambda)$, then the average fitness improvement is $\Omega(\log(\lambda)/\log\log(\lambda))$. Consequently, additive drift analysis (Theorem 4) tells us that only $O(n \log\log(\lambda)/\log(\lambda))$ iterations are needed to find a search point with fitness distance at most $n \log\log(\lambda)/\log(\lambda)$. Note that the fitness range from the typical initial fitness distance of $n/2$ to a fitness distance of $n \log\log(\lambda)/\log(\lambda)$ contains $\Omega(n)$ fitness levels. Hence the previous analysis would have given only a bound of $O(n)$ iterations.

There is an intuitive explanation for these numbers based on the balls-into-bins paradigm. When our current search point x is in distance n/D from the optimum, then, in the notation of Algorithm 1, already $x^{(1)}$ has an expected number of at least λ/D "good bits", i.e., bit positions that are zero in x and one in $x^{(1)}$. The same is true for x'. Each of these good bits is copied in each of the $y^{(j)}$ generated in the crossover phase with probability $1/\lambda$. The total number of copies of good bits in $y^{(1)}, \dots, y^{(\lambda)}$ thus is around λ/D again. Since they are uniformly spread over the $y^{(j)}$, we are in a situation closely resembling the *balls-into-bins* scenario, in which λ/D balls are uniformly thrown into λ bins. By a result of Raab and Steger [53], we know that when D is at most polylogarithmic in λ, then the most-loaded bin will contain $\Theta(\log(\lambda)/\log\log(\lambda))$ balls. For our setting, this means that we expect one of the $y^{(j)}$ to inherit $\Omega(\log(\lambda)/\log\log(\lambda))$ good bits. Unfortunately, since we do not distribute the good bits completely independently, we cannot transform this intuitive argument into a rigorous proof, but need to argue differently.

We start by analyzing the mutation phase. Since we aim at understanding those iterations where we gain more than a constant number of fitness levels, we restrict ourselves to the case that $\lambda = \omega(1)$, which eases the calculations.

Lemma 5 *Let $\varepsilon \in (0,1]$ be a constant. Assume that $\lambda = \omega(1)$. Let $x \in \{0,1\}^n$. Let $d := d(x) := n - \textsc{Om}(x)$ and $D := n/d$. Assume that $D = o(\lambda)$. Consider one run of the mutation phase of Algorithm 1. As in the description of the algorithm, denote by ℓ the actual mutation strength and by x' the winner individual. Let $B' := \{i \in [n] \mid x_i = 0 \wedge x'_i = 1\}$ the set of 1-bits that x' has gained over x.*

Then with probability $1 - o(1)$, we have both $|\ell - \lambda| \leq \varepsilon\lambda/2$ and $|B'| \geq (1-\varepsilon)\lambda/D$.

Proof Since $\lambda = \omega(1)$ and ℓ follows a binomial distribution with parameters n and λ/n, a simple application of the Chernoff bound (Theorem 1 (b) and (d)) implies that with probability $1 - o(1)$ we have $|\ell - \lambda| \leq (\varepsilon/2)\lambda$. Conditional on that, we analyze how the first offspring $x^{(1)}$ is generated. Let B_1 be the set of bit positions that are zero in x and one in $x^{(1)}$, that is, $B_1 := \{i \in [n] \mid x_i = 0 \wedge x_i^{(1)} = 1\}$ ("good bits"). Then $E[|B_1|] = d(\ell/n) = \ell/D$. Since $D = o(\lambda)$ and $\ell = \Theta(\lambda)$, this expectation is $\omega(1)$ and a Chernoff bound for the hypergeometric distribution (Theorems 3 and 1 (d)) shows that we have $\Pr[|B_1| \geq (1 - (\varepsilon/2))\ell/D] = 1 - o(1)$. Since all $x^{(j)}$, $j \in [\lambda]$, have the same Hamming distance from x, the fittest individual x' is also the one with the largest number of good bits. Hence $|B'| \geq |B_1| \geq (1-(\varepsilon/2))\ell/D \geq (1-\varepsilon)\lambda/D$ with probability $1 - o(1)$. □

We next analyze a run of the crossover phase. While in the previous lemma we only exploited that an individual generated in the mutation phase has roughly as many good bits as expected, we shall now exploit that the best of the λ individuals generated in the crossover phase is much better than the average one.

Lemma 6 *Let $x, x' \in \{0,1\}^n$ such that their Hamming distance $\ell := H(x, x')$ satisfies $\ell \leq 2\lambda - 2$. Let D' be such that $B' := \{i \in [n] \mid x_i = 0 \wedge x'_i = 1\}$ satisfies $|B'| \geq \lambda/D'$. Consider a run of the crossover phase starting with these variable values and computing an offspring $y \in \{0,1\}^n$.*

Then with probability at least $1 - 1/e$, we have

$$\textsc{Om}(y) - \textsc{Om}(x) \geq \lfloor \min\{(\tfrac{1}{2}\ln(\lambda) - 1)/(\ln\ln(\lambda) + \ln(D')), \lambda/D'\}\rfloor.$$

Proof Let $\gamma \leq \lambda/D'$ be a positive integer. Consider the outcome $y^{(j)}$ of a single crossover operation for some $j \in [\lambda]$ in Algorithm 1. Let A_j be the event that $\textsc{Om}(y^{(j)}) \geq \textsc{Om}(x) + \gamma$. This event in particular occurs when the crossover operation selects γ "good bits" (those with index in B') from x' and none of the "bad bits" (those, in which x and x' differ, but that are not in B'). Consequently,

$$\begin{aligned} \Pr[A_j] &\geq \binom{|B'|}{\gamma}(1/\lambda)^\gamma(1-1/\lambda)^{\ell-\gamma} \qquad (5)\\ &\geq (|B'|/\gamma)^\gamma\,(1/\lambda)^\gamma(1-1/\lambda)^{2(\lambda-1)}\\ &\geq (\lambda/(D'\gamma))^\gamma(1/\lambda)^\gamma(1/e^2)\\ &= \exp(-2-\gamma\ln\gamma-\gamma\ln D'). \end{aligned}$$

For $\gamma = \lfloor \min\{(\frac{1}{2}\ln(\lambda) - 1)/(\ln\ln(\lambda) + \ln(D')), \lambda/D'\}\rfloor$ we have $\Pr[A_j] \geq 1/\lambda$. Consequently, the probability that at least one of the A_j holds, is at least $1 - (1-1/\lambda)^\lambda \geq 1 - 1/e$. □

We note that the argument up to (5) is very similar to the reasoning in the proof of Theorem 5 in [41], where it is shown that the $(1 + \lambda)$ EA optimizing OneMax performs super-constant improvements in the early part of the optimization process. The choice of our γ, however, is different due to the different relation of λ and n. Interestingly, in the analysis of the mutation phase, such arguments do not seem to give significant additional improvement (recall that there we only used the expected gain from a fixed single offspring).

Above, we showed that in the early part of the optimization process, we regularly gain more than one fitness level in one iteration. For the remainder, we re-use the fitness level type argument of [16], which is summarized in the following lemma.

Lemma 7 (Lemma 7 of [16] for the special case that $k = \lambda$) *Assume $\lambda \geq 2$. In the notation of the $(1+(\lambda,\lambda))$ GA, the probability that one iteration produces a search point y that is strictly better than the parent x, is at least*

$$p_{d(x)} := C\left(1 - \left(\frac{n - d(x)}{n}\right)^{\lambda^2/2}\right),$$

where $C > 0$ is an absolute constant.

We are now ready to prove the main result of this section.

Proof (of Theorem 7) We regard the different regimes of the optimization process separately, since they need very different arguments. If $\lambda = \omega(1)$, then let $d_0 := n \ln\ln(\lambda)/\ln(\lambda)$, else let $d_0 := n$ (and there is no first phase).

First phase: From the random starting point to a solution x with $d(x) \leq d_0 := n \ln\ln(\lambda)/\ln(\lambda)$ in $O(n \log\log(\lambda)/\log(\lambda))$ iterations. Let T_0 be the maximum (taken over x) expected time needed to go from an initial search point x with $d(x) > d_0$ to a search point with d-value at most d_0. Let $\bar{D} = \ln(\lambda)/\ln\ln(\lambda)$. Let $x \in \{0,1\}^n$ be any search point with $d(x) > d_0 = n/\bar{D}$. By Lemma 5 and 6, we see that with probability at least $p = 1 - (1/e) - o(1)$, one iteration of the main loop of Algorithm 1 produces a solution y with $\textsc{Om}(y) \geq \textsc{Om}(x) + \Delta$, where Δ is some number satisfying $\Delta = \Omega(\log(\lambda)/\log\log(\lambda))$. This seems to call for an application of additive drift (Theorem 4), but in particular for the derivation of the large deviation claim, the following hand-made solution seems to be easier (despite several tail bounds for additive drift existing, see, e.g., [46] and the references therein).

For $t = 1, 2, \ldots$ let us define the following binary random variable X_t. If at the start of iteration t we have $d(x) > d_0$, then $X_t = 1$ if and only if $\textsc{Om}(y) \geq \textsc{Om}(x) + \Delta$. If $d(x) \leq d_0$, let $X_t = 1$ with probability p independent from all other random decisions. For all $t > 0$, we observe that $Y_t := \sum_{i=1}^{t} X_i \geq n/\Delta$ implies that $T_0 \leq t$, that is, our $(1 + (\lambda, \lambda))$ GA needed at most t iterations to find a search point x with $d(x) \leq d_0$. We have $E[Y_t] \geq tp$ and $\Pr[Y_t \leq (1/2)E[Y_t]] \leq \exp(-E[Y_t]/8)$. In particular, for $t = 2n/(\Delta p)$, we have $E[Y_t] \geq 2n/\Delta$ and $\Pr[Y_t \leq n/\Delta] \leq \exp(-E[Y_t]/8) = \exp(-n/(4\Delta)) = \exp(-n^{1-o(1)})$.

Second phase: From a solution with d-value at most d_0 to one with d-value at most $d_1 := \lfloor n/(2\lambda^2) \rfloor$ in $O(n \log\log(\lambda)/\log(\lambda))$ iterations. Once we have a solution of fitness distance at most d_0, we use the fitness level argument analogous to the proof of Theorem 6. We reformulate the proof slightly to allow proving a large deviation bound for the optimization time. By Lemma 7, the remaining number of iterations is dominated by a sum of geometric random variables $X_{d_0}, \ldots, X_1$ where $\Pr[X_d = m] = (1 - p_d)^{m-1} p_d$ for all $m = 1, 2, \ldots$ and $p_d = C\,(1 - (\frac{n - d(x)}{n})^{\lambda^2/2})$ is as in Lemma 7.

Note that for $d \geq d_1$, $p_d = C'$ for some absolute constant C'. Hence the expected number T_1 of iterations to reduce the fitness distance to d_1 is at most $E[T_1] = E[X_{d_0} + \cdots + X_{d_1 - 1}] \leq (1/C')(d_0 - d_1) \leq (1/C')d_0 = O(n \log\log(\lambda)/\log(\lambda))$ by linearity of expectation. Since each iteration with $d(x) \geq d_1$, independent of what happened in the previous iterations, has a success chance of at least C', we observe that the probability to have fewer than $(d_0 - d_1)$ successes in $2(1/C')(d_0 - d_1)$ iterations is at most $\exp(-(d_0 - d_1)/(4C')) = \exp(-\Theta(d_0)) = \exp(-n^{1-o(1)})$. Note that to apply the (multiplicative) Chernoff bound, here we used the "moderate independence" argument of Lemma 3.

Third phase: From a solution with fitness distance at most d_1 to an optimal solution in $O(n \log(n)/\lambda^2)$ iterations. We continue to use

the fitness level method as in the previous section of the proof, but note that for $d \leq d_1$, we have $p_d = C(1 - (1 - d/n)^{\lambda^2/2}) \geq C(1 - \exp(d\lambda^2/(2n))) \geq Cd\lambda^2/(4n)$, where we used the estimate $e^{-x} \leq 1 - x/2$ valid for all $x \in [0, 1]$. We thus see that the remaining time T_2 to get to the optimal solution is dominated by $X = X_{d_1} + \cdots + X_1$, which is a sum of independent geometric random variables with harmonic expectations. Hence by Lemma 4, we have $E[T_2] \leq E[X] \leq \frac{4n(\ln(d_1)+1)}{C\lambda^2} = O(n \log(n)/\lambda^2)$ and $\Pr[T_2 \geq (1 + \delta)\frac{4n \ln(d_1)}{C\lambda^2}] \leq d_1^{-\delta}$ for any $\delta > 0$.

In total, we see that the number $T = T_0 + T_1 + T_2$ of iterations until the optimum is found has an expectation of at most $E[T] = E[T_0]+E[T_1]+E[T_2] = O(\max\{n \log(n)/\lambda^2, n \log\log(\lambda)/\log(\lambda)\})$ and the probability that this upper bound is exceeded by a constant factor of $(1 + \delta)$ is only $O((n/\lambda^2)^{-\delta})$.

Since in each iteration the fitness of $O(\lambda)$ search points is computed, we proved the claimed upper bound of $O(\max\{n \log(n)/\lambda, n\lambda \log\log(\lambda)/\log(\lambda)\})$ for the expected optimization time, and again, exceeding this expectation by a factor of $1 + \delta$ has a probability of only $O((n/\lambda^2)^{-\delta})$. □

5 A Lower Bound for the Whole Parameter Space

As described in Section 2.1, a combination of intuitive considerations and rigorous work made in [16] and Section 4 suggest the parameter choice

$$\lambda = \lambda^* := \sqrt{\frac{\log(n) \log\log(n)}{\log\log\log(n)}},$$

$p^* = \lambda^*/n$, and $c^* = 1/\lambda^*$ for the optimization of the OneMax function class, yielding an expected optimization time of

$$F^* = \frac{n \log n}{\lambda^*} = n\sqrt{\frac{\log(n) \log\log\log(n)}{\log\log(n)}}.$$

As discussed right after Theorem 7 it was also proven in [14] that with p and c functionally depending on λ as above, $\lambda = \Theta(\lambda^*)$ is the optimal choice and the only optimal choice (this conclusion also follows from the following result).

In this section, we complete this picture by proving rigorously that no combination of the parameters p, c, and λ, all possibly depending on n, can lead to an expected optimization time of asymptotic order strictly better than F^*. We also show that not many parameter combinations can give this optimal expected runtime.

Theorem 8 *Let* $\lambda^* := \sqrt{\frac{\log(n) \log\log(n)}{\log\log\log(n)}}$ *and* $F^* = \frac{n \log n}{\lambda^*} = n\sqrt{\frac{\log(n) \log\log\log(n)}{\log\log(n)}}$.

(i) For arbitrary parameters $\lambda \in [0..n]$, $p \in [0, 1]$ *and* $c \in [0, 1]$, *all being functions in* n, *the* $(1 + (\lambda, \lambda))$ *GA has an expected optimization time of* $E[F] = \Omega(F^*)$.

(ii) If some parameter combination (λ, p, c) leads to an expected optimization time of $E[F] = \Theta(F^*)$, then
- $\lambda = \Theta(\lambda^*)$,
- $p = \Omega(\lambda^*/n)$ *and* $p = (1/n)\exp(O(\sqrt{\log(n)\log\log\log(n)/\log\log(n)}))$, *and*
- $c = \Theta(1/pn)$.

We remark that *the same lower bound holds for the natural modification of the $(1 + (\lambda, \lambda))$ GA in which the best of all mutation and crossover offspring competes in the final selection step with the parent individual* (and not only the best crossover offspring). The proofs below are written up in a way that this is easy to check, but to keep the paper readable we do not explicitly formulate all statements for both versions of the algorithm. Consequently, for the OneMax function, this modification does not give an asymptotic runtime improvement. In a practical application, however, there is no reason to not exploit possible exceptionally good mutation offspring. So here this modification seems very advisable. Recall that the upper bound proven in the previous section, as argued there, also holds for the modified algorithm.

To ease the presentation, we shall always parametrize the algorithm parameters by $p = k/n$ and $c = r/k$ for some $k \in (0, n]$ and $r \in [0, k]$ which may also depend on n. In this language, the previously suggested values are $k^* = \lambda^*$ and $r^* = 1$, and the main result of this section is that

(i) no parameter setting gives a better expected optimization time than the $\Theta(F^*)$ stemming from these parameters, and
(ii) any parameter tuple (λ, k, r) that leads to an asymptotic optimization time of $\Theta(F^*)$ satisfies
- $\lambda = \Theta(\lambda^*)$,
- $k = \Omega(k^*)$ and $k = \exp(O(\sqrt{\log(n)\log\log\log(n)/\log\log(n)}))$, and
- $r = \Theta(r^*)$.

A side remark: Another implicit parameter choice done in [16] is to use the same offspring population size λ for the mutation phase and the crossover phase. One could well imagine having different numbers λ_m and λ_c of offspring for both phases. This may make sense in practical applications or when performing a theoretical analysis that takes care of constant factors. In this work, where we are only precise up to the asymptotic order of magnitude, the expected optimization time is of asymptotic order equal to the product of the number of iterations and $\max\{\lambda_m, \lambda_c\}$. Hence, unless one believes that one can obtain a super-constant factor speed-up by reducing λ_m or λ_c, which is not what our proofs suggest, there is no advantage for us in not taking both offspring population sizes equal to $\max\{\lambda_m, \lambda_c\}$.

5.1 Overview of the Proof

Given the apparent difficulty (see Section 4) of computing the expected runtime of the $(1 + (\lambda, \lambda))$ GA already for settings $k = \lambda$ and $r = 1$ suggested

in [16], the common approach of determining the optimal parameter settings by conducting a precise runtime analysis for all parameter combinations (λ, k, r) seems not very promising. Therefore, we shall rather analyze particular parts of the optimization process in detail and from these extract necessary conditions for the parameters to allow an expected optimization time of order $O(F^*)$. To make it more visible how the different arguments work together, let us start with a brief overview of the analysis.

Let a tuple $(\lambda, p = k/n, c = r/k)$ of the parameters be given. We denote by T the number of *iterations* (!) that the $(1 + (\lambda, \lambda))$ GA with these parameters performs until an optimal solution is found for the first time (we have $T = 0$ if the random initial search point is already optimal). We denote by F the optimization time of this $(1 + (\lambda, \lambda))$ GA, that is, the number of *fitness evaluations* performed until an optimal solution is evaluated. F equals one if the random initial search point is optimal. Roughly it holds that $F \approx 2\lambda T$, but see Proposition 3 and the text around it for the details.

We say that a tuple of parameters is *optimal* if the resulting expected optimization time is $O(F^*)$. This is, for the moment, a slight abuse of language, but as this section will show, these are indeed the parameters that lead to the asymptotically optimal expected runtime, since (as we will see) no better expected runtime than $\Omega(F^*)$ can be achieved with any parameter setting. The proof of Theorem 8 then consists of the following arguments, which can all be shown independently of the others. Since we aim at an asymptotic result only, we can freely assume that n is sufficiently large.

- In Lemma 8, we make the elementary observation that $E[F] \geq \min\{\lambda, 2^n\}/2$. Consequently, $\lambda \leq 2F^*$ in any optimal parameter set.
- In Lemma 9, we show that

$$E[F] = \min\left\{\Omega\left(r^{-1}\exp(\Theta(r))n\log n\right), \exp(\Omega(r))n^2\log n, \exp(\Omega(n^{1/16}))\right\}$$

  when $k \geq \sqrt{n}$ and $\lambda = \exp(o(n^{1/16}))$. Since this runtime is at least $\Omega(n \log n)$, together with the previous item (showing that λ cannot be too large), we obtain that, in an optimal parameter set, k is at most $\sqrt{n}$.
- In Lemma 10, we show that for $0 < k \leq n/12$, we have $E[F] = \Omega(\frac{n \log n}{k})$. Hence $k = \Omega(\lambda^*)$ in an optimal parameter setting.
- In Lemma 12, we show that when $\omega(1) = k \leq \sqrt{n}$, then

$$E[F] = \Omega\left(n \log n \min\left\{\tfrac{\exp(\Omega(r))}{\lambda r}, \tfrac{n^3}{\lambda}, \tfrac{\exp(\Omega(k))}{k}\right\}\right).$$

  Since we know already that $\lambda \leq n^3$ and $k = \omega(1)$ in an optimal parameter setting, this result implies that an optimal parameter set has $\lambda = \Omega(\lambda^* \exp(\Omega(r))/r)$.
- In Lemma 13, we show $E[F] = \Omega(n\lambda/k)$ when $k \leq n/4$ (which we know already). Consequently, in an optimal set of parameters λ cannot be excessively large, e.g., $\lambda \leq \exp(k/120)$.

– In Lemma 14, we show that if $k \leq n/80$, $\lambda \leq \exp(k/120)$, $\lambda = \exp(o(n))$, and $\lambda = \omega(1)$—all of this holds in an optimal parameter setting as shown above—then $E[F] = \Omega(\frac{n\lambda \log\log(\lambda)}{r \log \lambda})$. This result together with Lemma 12 implies that the optimal expected runtime is $\Theta(F^*)$ and that we have $\lambda = \Theta(\lambda^*)$ and $r = \Theta(1)$ in an optimal parameter setting.

This shows the main claim of this section, namely that F^* is asymptotically the best expected runtime one can achieve with a clever choice of all parameters of the $(1 + (\lambda, \lambda))$ GA. The above also shows that an optimal parameter set has $\lambda = \Theta(\lambda^*)$ and $r = \Theta(1)$. For the mutation probability, the above only yields $k = \Omega(\lambda^*)$ and $k = O(\sqrt{n})$. In Lemma 15, we show that $k = \exp(O(\sqrt{\log(n) \log\log\log(n)/\log\log(n)}))$ is a necessary condition for having a $\Theta(F^*)$ expected runtime.

We do not know if the interval of optimal values for k can be further reduced. An inspection of the proof of the upper bound presented in Section 4 suggests that, with more effort than there, also slightly larger k-values than $\Theta(\lambda^*)$ (together with $\lambda = \Theta(\lambda^*)$ and $r = \Theta(1)$) could lead to the optimal expected runtime of $\Theta(F^*)$. We do not follow up on this question, because we do not feel that it justifies the effort of extending the technical proof of Section 4. It is quite clear that there is no algorithmic advantage of using a larger than necessary k-value. The main (unfavorable) difference would be that than an efficient implementation of the mutation operator in expected time $\Theta(k)$ would have an increased complexity.

We face two main difficulties in the proof of Theorem 8. One are the apparent dependencies introduced by the two intermediate selection steps and the fact that all mutation offspring have the same Hamming distance from the parent. That the latter creates additional challenges can be easily seen in the lengthy proof of Lemma 10, which simply tries to use the classic argument that one needs at least a total number of $\Theta(n \log n)$ bit-flips to make sure that each initially incorrect bit was flipped at least once.

The second difficulty is that even parameter combinations that are far from those leading to the optimal expected runtime can lead to runtimes very close to the optimal one. An example (given here without proof) is that for say $k = \sqrt{n}$ and $\lambda = \lambda^*$ and $r = 1$, the optimization process strongly resembles the one of the $(1 + \lambda)$ EA with λ below the cut-off point. Consequently, the $(1 + (\lambda, \lambda))$ GA for these parameters has an expected optimization time of $\Theta(n \log n)$, which is relatively close to F^* despite the uncommonly large mutation probability.

5.2 Proofs

In this longer subsection, we prove the results outlined above. We frequently use the following notation. For $x \in \{0,1\}^n$, we call $d(x) := n - \textsc{Om}(x)$ its *fitness distance*. Let $x, x', y \in \{0,1\}^n$. Then

$$g(x, x') := |\{i \in [n] \mid x_i = 0 \wedge x'_i = 1\}|$$

is the number of *good bits of x' (with respect to x)*. Analogously,

$$b(x, x') := |\{i \in [n] \mid x_i = 1 \wedge x'_i = 0\}|$$

is the number of *bad bits of x' (with respect to x)*. Note that, trivially, $g(x, x') + b(x, x') = H(x, x')$, the Hamming distance of x and x'. Similarly, we define "the number of good bits of x' that made it into y" and "the number of bad bits of x' that made it into y" by

$$g(x, x', y) := |\{i \in [n] \mid x_i = 0 \wedge x'_i = 1 \wedge y_i = 1\}|,$$
$$b(x, x', y) := |\{i \in [n] \mid x_i = 1 \wedge x'_i = 0 \wedge y_i = 0\}|,$$

respectively.

We remind the reader that in the following, we always assume that we consider a run of the $(1 + (\lambda, \lambda))$ GA with the general parameter setting λ, $p = k/n$, and $c = r/k$, which may all depend on the problem size n. Since we are interested in an asymptotic result, we may assume that n is sufficiently large. We use the variables of the algorithm description, e.g., x, $x^{(i)}$, x', etc. without further explicit reference to the algorithm (Algorithm 1).

We now prove the ingredients forming the proof of the main result. We prove these results not only for the minimal parameter range needed in the proof of the main result, but rather for those ranges where the main arguments work well. At the same time, we do not aim at the absolutely widest parameter range and we occasionally do not aim at the sharpest possible bound if this would significantly increase the proof complexity. We aim at keeping the proofs of the partial results independent, both to ease reading and to allow an easier understanding of how the main proof decomposes into the partial results. For this reason, all of the following lemmas are proven independently apart from possibly relying on the two elementary Propositions 1 and 3.

The first of these proposition is a technical tool showing that extraordinarily large fitness gains occur rarely. This allows in the following to assume that the algorithm indeed has, at some point in time, a parent individual x with roughly a certain fitness.

Proposition 1 *Let x be a search point with $d := d(x)$ satisfying $d \leq 0.6n$. Then the probability that one iteration of the $(1 + (\lambda, \lambda))$ GA with arbitrary parameter settings creates a search point y (as mutation or crossover offspring) with $d(y) \leq d/2$, is $\lambda(\lambda + 1) \exp(-\Omega(d))$.*

To prove this proposition, we need the elementary fact that standard bit mutation hardly reduces $d(\cdot)$ by 50% or more.

Proposition 2 *Let $p \in [0, 1]$, $x \in \{0, 1\}^n$ with $d := d(x) \leq 0.6n$, and let y be obtained from flipping each bit of x independently with probability p. Then $\Pr[d(y) \leq 0.5d] = \exp(-\Omega(d))$.*

Proof Let first $0.1n \leq d \leq 0.6n$. Then $E[d(y)] \geq \min\{d, 0.4n\}$ regardless of p. Consequently, $\Pr[d(y) \leq d/2] \leq \Pr[d(y) \leq E[d(y)] - 0.05n] \leq \exp(-\Theta(n))$ by the additive Chernoff bound (Theorem 1 (e)).

Let now $d \leq 0.1n$. Let $g := g(x,y)$ and $b = b(x,y)$. Trivially, we have $d(y) = d - g + b$. Let first $p \leq 1/4$. Since g is binomially distributed with parameters d and p, we have $E[g] = dp \leq d/4$ and $\Pr[g \geq d/2] \leq \exp(-\Omega(d))$ by the multiplicative Chernoff bound (Theorem 1 (d)). We thus have $\Pr[d(y) \leq d/2] \leq \Pr[g \geq d/2] \leq \exp(-\Omega(d))$. Let now $p \geq 1/4$. Then $E[b] = (n-d)p \geq 0.225n$ and $\Pr[b \leq 0.1n] \leq \exp(-\Omega(n))$. Since trivially $g \leq d \leq 0.1n$, we have $\Pr[d(y) \leq d/2] \leq \Pr[b \leq 0.1n] \leq \exp(-\Omega(n))$. □

Proof (of Proposition 1) To ease the calculations, we use the following gedankenexperiment. Imagine that the $(1+(\lambda,\lambda))$ GA does not select a winning individual x' at the end of the mutation phase, but instead creates λ crossover offspring from each of the λ mutation offspring. Clearly, the set of λ crossover offspring from a true run of the algorithm is contained in this set of λ^2 offspring. Hence it suffices to show that none of the λ^2 offspring from the Gedankenexperiment and none of the λ mutation offspring has a fitness distance of $d/2$ or better.

Let $\tilde{y}$ be a crossover offspring of the Gedankenexperiment. Let $\tilde{x}$ be the mutation offspring that was used in the crossover giving rise to $\tilde{y}$. Then $\tilde{x}$ is obtained from x by flipping each bit independently with probability k/n—the $(1+(\lambda,\lambda))$ GA creates $\tilde{x}$ algorithmically different, namely by first sampling ℓ and then flipping ℓ bits, but the result is that $\tilde{x}$ has the distribution described above due to the choice of ℓ. Now $\tilde{y}$ is obtained from a biased crossover of x and $\tilde{x}$. Since each bit of $\tilde{x}$ makes it into $\tilde{y}$ only with probability r/k, we see that we have $\tilde{y}_i \neq x_i$ with probability $(k/n) \cdot (r/k) = r/n$ independently for all $i \in [n]$. Consequently, $\tilde{y}$ has the same distribution as if it was generated from x by standard bit mutation with mutation rate r/n.

Since all mutation and crossover offspring are distributed as if generated via standard bit mutation (with some mutation rate that does not matter here), Proposition 2 and a simple union bound over the $\lambda(\lambda+1)$ mutation and crossover offspring shows that with probability at least $1-\lambda(\lambda+1)\exp(-\Omega(d))$ none of these has a fitness distance of $d/2$ or better. □

The following proposition shows that, apart from exceptional cases, we can freely switch between the number of iterations T and the number of fitness evaluations F needed to find an optimum. This is a well-known fact, so we present its proof merely for reasons of completeness. Recall that the optimization time is defined to be the number of fitness evaluations until an optimal solution is evaluated for the first time. Consequently, if, say, the first mutation offspring by chance is an optimal solution, then the optimization time F would be 2. The number of iterations T, though, would be 1, so the estimate $F = \Omega(\lambda T)$ is not valid. The following lemma shows this exceptional case only occurs for $E[T] < 2$, so that usually we can (and will without further notice) use the argument $E[F] = \Omega(\lambda E[T])$.

Proposition 3 *If $E[T] \geq 2$, then $E[F] = \Theta(\lambda E[T])$.*

Proof By definition of F and T, we have $T = \lceil (F-1)/2\lambda \rceil \leq (F-1)/2\lambda + 1$. Consequently, $F \geq 2(T-1)\lambda+1$ and $E[F] \geq E[2(T-1)\lambda+1] \geq 2(E[T]-1)\lambda \geq E[T]\lambda$ when $E[T] \geq 2$. Since $F \leq 2\lambda T + 1$, we also have $E[F] = O(\lambda E[T])$. □

We now start proving a number of lower bounds for the runtime of the $(1 + (\lambda, \lambda))$ GA. They do not logically rely on each other. The first result shows that, unless λ is excessively large, the expected optimization time is at least $\Omega(\lambda)$. This follows from observing that each of the mutation offspring in the first iteration is uniformly distributed in the search space, and hence, has a very small probability of being equal to the optimal solution.

Lemma 8 $E[F] \geq \min\{\lambda, 2^n\}/2$.

Proof The proof builds on the following simple observation: Let $\tilde{x}$ be a mutation offspring generated in the first iteration. Then $\tilde{x}$ is uniformly distributed in $\{0,1\}^n$. Indeed, let x be the random initial search point, which is uniformly distributed in $\{0,1\}^n$, which is equivalent to saying that each x_i independently is equal to 1 with probability $1/2$ (and is equal to 0 otherwise). Now $\tilde{x}$ is generated from x by flipping each bit independently with probability k/n. Consequently, the bits of $\tilde{x}$ are independent. We also compute $\Pr[\tilde{x}_i = 1] = \Pr[x_i = 0](k/n) + \Pr[x_i = 1](1 - k/n) = 1/2$. Hence $\tilde{x}$ is uniformly distributed in $\{0,1\}^n$.

With this preliminary consideration, the proof of the lemma is very easy. Let L be a non-negative integer. Let $x_0, x_1, \ldots, x_L$ be the initial random search point and the first L mutation offspring. Note that each of these search points individually is uniformly distributed in $\{0,1\}^n$. Consequently, by a simple union bound, the probability that one of these search points is the optimum is at most $(L+1)2^{-n}$. In other words, the number F of fitness evaluations until an optimal solution is found, satisfies $\Pr[F \geq L+2] \geq 1 - (L+1)2^{-n}$ for all $0 \leq L \leq \lambda$. By Lemma 2, taking $K = \min\{\lambda+1, 2^n\}$, we compute

$$\begin{aligned} E[F] &= \sum_{i=1}^{\infty} \Pr[F \geq i] \geq \sum_{i=1}^{K} \Pr[F \geq i] \geq \sum_{i=1}^{K} (1 - (i-1)2^{-n}) \\ &= K - \frac{K(K-1)}{2} 2^{-n} = K(1 - 2^{-n-1}(K-1)) \geq \min\{\lambda, 2^n\}/2. \end{aligned}$$

□

We proceed by regarding the case that k is large, say $k \geq \sqrt{n}$. Despite the fact that this is much larger than all values of k that lead to the optimal expected runtime, the proof is not very simple. The reason is that even such large values for k can give a near-optimal expected runtime of $O(n \log n)$ for suitable choices of the other parameters, e.g., small values for λ and $r = 1$ (we do not prove this statement). The main intuitive reason for the following lemma to be true is that for k large and d fairly large, all mutation offspring contain very

similar numbers of good and bad bits. Consequently, we do not gain anything from generating many mutation offspring in parallel and selecting the best one for the crossover phase.

Lemma 9 *If $k \geq \sqrt{n}$ and $\lambda = \exp(o(n^{1/16}))$, then*

$$E[F] = \min\left\{\Omega\left(r^{-1}\exp(\Theta(r))n\log n\right), \exp(\Omega(r))n^2\log n, \exp(\Omega(n^{1/16}))\right\},$$

which attains its asymptotically optimal value $\Omega(n\log n)$ for $r = \Theta(1)$.

Proof We start by analyzing the progress the $(1+(\lambda,\lambda))$ GA makes in one iteration starting with a search point x having fitness distance $d := d(x) \in [n^{3/4}, n^{7/8}]$. More precisely, denote by z an individual with maximal fitness among all mutation and crossover offspring generated in this iteration and among the parent x. Needless to say, z can be the parent x, the crossover winner y, or the mutation winner x'. To use drift analysis, we shall regard the progress $d(x) - d(z)$. Note that this is 0 if $\max\{\textsc{Om}(x'), \textsc{Om}(y)\} \leq \textsc{Om}(x)$. Note also that $d(x) - d(z) = \textsc{Om}(z) - \textsc{Om}(x)$.

Let $\tilde{x}$ be a mutation offspring. Let $\tilde{g} = g(x,\tilde{x})$ be the number of good bits of $\tilde{x}$. Since $\tilde{g}$ follows a binomial distribution with parameters d and k/n, we have $E[\tilde{g}] = dk/n \geq n^{1/4}$ and $\Pr[\tilde{g} \geq 2dk/n] \leq \exp(-(dk/n)/3) \leq \exp(-\Omega(n^{1/4}))$. Hence only with probability at most $\lambda\exp(-\Omega(n^{1/4}))$, there is a mutation offspring with at least $2dk/n$ good bits; in this rare case we estimate the progress $\textsc{Om}(z) - \textsc{Om}(x)$ via the trivial bound $\textsc{Om}(z) - \textsc{Om}(x) \leq n$. Similarly, in the exceptional case that $\ell < k/2$, which occurs with probability at most $\exp(-\Omega(k)) \leq \exp(-\Omega(n^{1/2}))$, we again estimate $\textsc{Om}(z) - \textsc{Om}(x) \leq n$.

Hence let us now analyze the progress in the *regular situation* that no mutation offspring has $2dk/n$ good bits or more (and thus $g(x,x') < 2dk/n$) and that $\ell \geq k/2$ (and thus x' has at least $b(x,x') \geq \ell - (2dk/n) \geq (k/2) - (2dk/n) = k((1/2) - 2n^{-1/8}) \geq k/4$ bad bits). Since $b(x,x') > g(x,x')$, we have $z \neq x'$, so it remains to analyze the crossover offspring. Consider an offspring $\tilde{y}$ generated in the crossover phase.

Let us consider first *the case that $r \geq n^{1/16}$*. Then $\tilde{b} := b(x,x',\tilde{y})$ satisfies $E[\tilde{b}] \geq (k/4)\cdot(r/k) = r/4$. Hence with probability $1 - \exp(-\Omega(r)) \geq 1 - \exp(-\Omega(n^{1/16}))$, the crossover offspring $\tilde{y}$ has taken at least $k/8$ bad bits from x'. This is more than the number of good bits x' has, so regardless of how many good bits make it into $\tilde{y}$, we have $\textsc{Om}(\tilde{y}) \leq \textsc{Om}(x)$. Consequently, with probability $1 - \lambda\exp(-\Omega(n^{1/16}))$, no crossover offspring has a fitness better that x, and hence $\textsc{Om}(z) = \textsc{Om}(x)$. For the remaining probability $\lambda\exp(-\Omega(n^{1/16}))$, we estimate $\textsc{Om}(z) - \textsc{Om}(x) \leq n$. In total, if $r \geq n^{1/16}$, we have $E[\textsc{Om}(z) - \textsc{Om}(x)] \leq n\lambda\exp(-\Omega(n^{1/16})) = \lambda\exp(-\Omega(n^{1/16}))$.

We now turn to *the case that $r < n^{1/16}$*. In this case, $\tilde{g} := g(x,x',\tilde{y})$ satisfies $E[\tilde{g}] \leq (2dk/n)\cdot(r/k) = 2dr/n \leq 2n^{-1/16}$. We regard separately the situations that $\tilde{g} = 0$, $\tilde{g} \in [1..47]$, $\tilde{g} \in [48..\lfloor E[\tilde{b}]/2\rfloor]$, and $\tilde{g} \geq E[\tilde{b}]/2$. Clearly, when $\tilde{g} = 0$, we have $\textsc{Om}(\tilde{y}) \leq \textsc{Om}(x)$. Markov's inequality shows that good bits exist only with probability $E[\tilde{g}] \leq 2dr/n$, hence, $\Pr[\tilde{g} \in [1..47]] \leq \Pr[\tilde{g} \geq 1] \leq 2dr/n$. Conditioning on $\tilde{y}$ having between one and 47 good bits, we trivially

observe $\textsc{Om}(\tilde{y}) - \textsc{Om}(x) \leq 47$. However, for $\tilde{y}$ to have a fitness better than $\textsc{Om}(x)$, it is necessary (but not sufficient) that at most 46 bad bits are copied from x' to $\tilde{y}$. The probability of this event, which is independent of any event regarding good bits only, is at most $\exp(-\Omega(E[\tilde{b}])) \leq \exp(-\Theta(r))$, because the expected number $E[\tilde{b}]$ of bad bits copied into $\tilde{y}$ is $\Theta(r)$. By Theorem 1 (a), the probability that 48 or more good bits are copied into $\tilde{y}$ is $O(n^{-3})$, hence $\Pr[\tilde{g} \in [48..\lfloor E[\tilde{b}]/2 \rfloor]] \leq \Pr[\tilde{g} \geq 48] = O(n^{-3})$. In this situation, for $\textsc{Om}(\tilde{y})$ to be larger than $\textsc{Om}(x)$, we need $\tilde{b} < \tilde{g} \leq E[\tilde{b}]/2$, which happens with probability $\exp(-\Omega(E[\tilde{b}])) \leq \exp(-\Omega(r))$. Finally, if $E[\tilde{b}]/2 \geq 48$, then the probability that $\tilde{g} \geq E[\tilde{b}]/2$ is at most $n^{-3-\Omega(E[\tilde{b}])} \leq n^{-3-\Omega(r)}$ by Theorem 1 (a). Hence

$$\begin{aligned}
&E[\max\{\textsc{Om}(\tilde{y}) - \textsc{Om}(x), 0\}] \\
&\quad \leq \Pr[\tilde{g} = 0] \cdot 0 + \Pr[\tilde{g} \in [1..47]] \exp(-\Omega(r)) \cdot 47 \\
&\qquad + \Pr[\tilde{g} \in [48..\lfloor E[\tilde{b}]/2 \rfloor]] \exp(-\Omega(r)) \cdot n \\
&\qquad + \Pr[\tilde{g} \geq E[\tilde{b}]/2 \mid E[\tilde{b}]/2 \geq 48] \cdot n \\
&\quad \leq 0 + \tfrac{2dr}{n} \exp(-\Omega(r)) \cdot 47 + O(n^{-3}) \exp(-\Omega(r)) \cdot n + n^{-3-\Omega(r)} \cdot n \\
&\quad \leq O\left((\tfrac{dr}{n} + n^{-2}) \exp(-\Omega(r)) \right).
\end{aligned}$$

Since y is chosen among the crossover offspring $\tilde{y}$ such that $\textsc{Om}(\tilde{y})$, and equivalently, $\textsc{Om}(\tilde{y}) - \textsc{Om}(x)$ is maximal, we have $\textsc{Om}(y) - \textsc{Om}(x) \leq \sum_{\tilde{y}} \max\{\textsc{Om}(\tilde{y}) - \textsc{Om}(x), 0\}$, where $\tilde{y}$ runs over all λ crossover offspring. Consequently, $E[\textsc{Om}(y) - \textsc{Om}(x)] = O(\lambda(\tfrac{dr}{n} + n^{-2}) \exp(-\Omega(r)))$.

Taking the two cases regarded separately together, we see that for any r we have

$$\begin{aligned}
E[\textsc{Om}(z) - \textsc{Om}(x)] &= E[\max\{0, \textsc{Om}(y) - \textsc{Om}(x)\}] \\
&= \max\left\{ O\left(\lambda(\tfrac{dr}{n} + n^{-2}) \exp(-\Omega(r)) \right), \lambda \exp(-\Omega(n^{1/16})) \right\},
\end{aligned}$$

when we condition on being in the regular situation. In the general situation, we have

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x)] \\
&\quad = \lambda \exp(-\Omega(n^{1/4}))n + (1 - \lambda \exp(-\Omega(n^{1/4}))) \cdot \\
&\qquad \max\left\{ O\left(\lambda(\tfrac{dr}{n} + n^{-2}) \exp(-\Omega(r)) \right), \lambda \exp(-\Omega(n^{1/16})) \right\} \\
&\quad = \max\left\{ O\left(\lambda(\tfrac{dr}{n} + n^{-2}) \exp(-\Omega(r)) \right), \lambda \exp(-\Omega(n^{1/16})) \right\}.
\end{aligned}$$

To ease the following multiplicative drift argument, we estimate this bluntly by

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x)] \\
&\quad \leq \max\left\{ O(\lambda(\tfrac{dr}{n} + dn^{-2}) \exp(-\Omega(r))), d\lambda \exp(-\Omega(n^{1/16})) \right\} \\
&\quad = d \max\left\{ O\left(\lambda \max\{r, n^{-1}\} \exp(-\Omega(r))/n \right), \lambda \exp(-\Omega(n^{1/16})) \right\}.
\end{aligned}$$

Building on this drift statement, we now use Witt's lower bound result for multiplicative drift (Theorem 5). Consider a run of the $(1 + (\lambda, \lambda))$ GA. For $t = 0, 1, \dots$, denote by x_t the search point x at the beginning of the $(t + 1)$st iteration; except if before the $(t + 1)$st iteration an optimal solution has been evaluated, in which case we let x_t be any optimal solution. By Proposition 1, with probability at least $1-\lambda(\lambda+1)\exp(-\Omega(n^{7/8}))$, the $(1+(\lambda, \lambda))$ GA at some time t_0 reaches a search point x_{t_0} with $d(x_{t_0}) \in [0.5n^{7/8}, n^{7/8}]$. We show that in this case, we have an expected optimization time as claimed, which implies that also the unconditioned expectation is of the same order of magnitude.

For $t = 0, 1, \dots$ define $X_t := \max\{d(x_{t_0+t}), 1\}$. Observe that $X_{t+1} \leq X_t$ for all $t \geq 0$. Let $s_{\min} := n^{3/4}$. Then we have shown above that if $X_t = s > s_{\min}$, then

$$E[X_t - X_{t+1}] \leq s \max\left\{K_1 \lambda \max\{r, n^{-1}\} \exp(-K_2 r)/n, \lambda \exp(-K_3 n^{1/16})\right\}$$

for some absolute constants K_1, K_2, K_3. Note that the drift of the process X_t might be smaller than this, because above we took z as the best individual among parent and all individuals generated in the iteration. The first condition of the drift theorem thus is fulfilled with $\delta = \max\{K_1 \lambda \max\{r, n^{-1}\} \exp(-K_2 r)/n, \lambda \exp(-K_3 n^{1/16})\}$. From Proposition 1 we know that $\Pr[X_{t+1} \leq s/2] \leq \lambda(\lambda + 1) \exp(-\Omega(s)) = \exp(-\Omega(s))$. Hence for n (and thus also s) sufficiently large, also the second condition of the drift theorem is satisfied (with $\beta = 1/2$); also we have $E[T] = \Omega(\log n)$ to enable the argument $E[F] = \Omega(\lambda E[T])$ below. We may thus apply the theorem and derive that the first t such that $X_t \leq s_{\min}$ satisfies

$$\begin{aligned} E[t] &= \Omega\left(\frac{\ln(X_0) - \ln(s_{\min})}{\delta}\right) \\ &= \Omega\left(\min\left\{\frac{\exp(\Theta(r) n \log n)}{\max\{r, 1/n\}\lambda}, \frac{\exp(\Omega(n^{1/16}))}{\lambda}\right\}\right). \end{aligned}$$

Note that this, naturally, is a lower bound on $E[T]$. Consequently,

$$\begin{aligned} E[F] &= \Omega(\lambda E[T]) \\ &= \Omega\left(\min\left\{\frac{\exp(\Omega(r))}{r} n \log n, \exp(\Omega(r)) n^2 \log n, \exp(\Omega(n^{1/16}))\right\}\right). \end{aligned}$$

□

The following lower bound imitates the classic argument that if in all applications of the mutation operator not enough bits are flipped, then there will be a bit that is initially zero and that has never been touched in any mutation operation. The proof is slightly more involved as usual for this type of argument because our mutation operator uses a hypergeometric distribution.

Lemma 10 *Let $0 < k \leq n/12$ and $k\lambda = o(n \log n)$. Let $\alpha < 1/4$. Let $t = \lfloor \alpha n \ln(n)/(k\lambda) \rfloor$. Then $\Pr[T \leq t] = \exp(-\Omega(\min\{kt, n^{1-4\alpha}\}))$. In particular, $E[F] = \Omega(\frac{n \log n}{k})$. Consequently, an optimal parameter setting satisfies $k = \Omega(\sqrt{\log(n) \log\log(n) / \log\log\log(n)}) = \Omega(\lambda^*)$.*

Proof Using the Chernoff bound of Theorem 1 (d), we see that with probability $1 - \exp(-\Omega(n))$, the initial search point has at least $n/3$ bits valued zero ("missing bits").

Let us consider what happens in the first $t = \lfloor \alpha n \ln(n)/(k\lambda) \rfloor$ iterations. Denote by $\ell_1, \dots, \ell_t$ the values of ℓ chosen by the algorithm in these iterations. Note that the ℓ_i are independent random variables each having a binomial distribution with parameters n and k/n. Consequently, $L := \sum_{i=1}^{t} \ell_i$ is a sum of tn independent $0,1$ random variables that are one with probability k/n. Hence we have $E[L] = tk$. By the multiplicative Chernoff bound of Theorem 1 (b), we see that with probability $1 - \exp(-\Omega(tk))$, we have $L \leq 2tk$.

Again exploiting the binomial distribution of the ℓ_i, we derive from Theorem 1 (c) that $\Pr[\ell_i \geq n/2] \leq 2^{-n/2}$; note that here we used that $k \leq n/12$ and thus $E[\ell_i] = k \leq n/12$. Consequently, with probability $1 - \exp(-\Omega(n))$, all ℓ_i are at most $n/2$ (union bound).

In the following, we condition on none of these three rare events occurring. More precisely, we condition on that there are at least $n/3$ missing bits and we condition on a particular outcome of the ℓ_i that avoids the exceptional events $L > 2tk$ and $\ell_i > n/2$ for some $i \in [t]$. The probability that a particular one of the missing bits is never flipped in the mutation phases of the first t iterations is

$$\begin{aligned}\prod_{i=1}^{t}(1-\ell_i/n)^\lambda &\geq \prod_{i=1}^{t}\exp(-2\ell_i/n)^\lambda = \exp(-2\lambda L/n) \geq \exp(-4k\lambda t/n)\\ &\geq n^{-4\alpha},\end{aligned}$$

where we have used in the first step that $1 - c \geq e^{-2c}$ for $0 \leq c \leq 1/2$.

Denote by $M \subseteq [n]$ the set of missing bits and by A_i the event that bit i was flipped at least once in some mutation step in the first t iterations. Then we just showed $\Pr[A_i] \leq 1 - n^{-4\alpha}$. We want to show that it is very unlikely that all events A_i are fulfilled.

Unfortunately, the events $A_i, i \in M$, are not independent, since already in a single application of the mutation operator the bits are not treated independently, but according to a hypergeometric distribution. We therefore now show that they satisfy the following negative correlation property:

$$\forall I \subseteq M : \Pr\left[\bigcap_{i\in I} A_i\right] \leq \prod_{i\in I} \Pr[A_i].$$

We proceed via induction over the cardinality of I. For $|I| = 0, 1$, there is nothing to show. Let $I \subseteq M$ such that $|I| \geq 2$. Let $j \in I$ and $I' := I \setminus \{j\}$. Then

$$\Pr\left[\bigcap_{i\in I'} A_i\right] = \Pr\left[\bigcap_{i\in I'} A_i \middle| A_j\right] \Pr[A_j] + \Pr\left[\bigcap_{i\in I'} A_i \middle| \bar{A}_j\right] \Pr[\bar{A}_j]. \tag{6}$$

It is clear that $\Pr[\bigcap_{i\in I'} A_i \mid \bar{A}_j]$ is at least as large as $\Pr[\bigcap_{i\in I'} A_i]$—conditioning on $\bar{A}_j$ is equivalent to saying that the random subsets of bits to

be flipped are not chosen as subsets of $[n]$, but of $[n] \setminus \{j\}$, and this increases the probability of the event $\bigcap_{i \in I'} A_i$. More formally, there is the following coupling from the unconditioned probability space into the one conditional on $\bar{A}_j$. Whenever in the unconditioned probability space the j-th bit is flipped in some iteration, we replace this bit-flip by flipping a new bit different from j and the other bits flipped in this iteration. This is exactly the random experiment done in the probability space conditional on $\bar{A}_j$. Clearly, if the event $\bigcap_{i \in I'} A_i$ holds in the unconditioned space, this is not affected by the coupling. Hence the probability of the event $\bigcap_{i \in I'} A_i$ is not smaller in the space conditional on $\bar{A}_j$.

Since thus $\Pr[\bigcap_{i \in I'} A_i \mid \bar{A}_j] \geq \Pr[\bigcap_{i \in I'} A_i]$, we see from equation (6) that $\Pr[\bigcap_{i \in I'} A_i \mid A_j] \leq \Pr[\bigcap_{i \in I'} A_i]$. From $\Pr[\bigcap_{i \in I'} A_i \mid A_j] = \Pr[\bigcap_{i \in I} A_i] / \Pr[A_j]$ we derive the desired statement $\Pr[\bigcap_{i \in I} A_i] \leq \Pr[\bigcap_{i \in I'} A_i] \Pr[A_j]$. Applying induction to I', we have

$$\Pr\left[\bigcap_{i \in I} A_i\right] \leq \prod_{i \in I'} \Pr[A_i] \Pr[A_j] = \prod_{i \in I} \Pr[A_i].$$

Using this negative correlation property for the set of all missing bits, we conclude that the probability $\Pr[\bigcap_{i \in M} A_i]$ that all missing bits were flipped at least once, is $\Pr[\bigcap_{i \in M} A_i] \leq \prod_{i \in M} \Pr[A_i] \leq (1 - n^{-4\alpha})^{n/3} \leq \exp(-n^{-4\alpha})^{n/3} = \exp(-n^{1-4\alpha}/3)$, where we used the estimate $(1 + x) \leq e^x$ valid for all $x \in \mathbb{R}$.

Consequently, with probability at least $1 - \exp(-\Omega(n)) - \exp(-\Omega(kt)) - \exp(-\Omega(n^{1-4\alpha}))$, there is a bit that initially has the value zero and is not flipped in the first t iterations, implying that the $(1 + (\lambda, \lambda))$ GA needs more than t iterations to generate the optimum as mutation or crossover offspring. This high-probability statement immediately implies the claimed bound on the expected optimization time, using again $E[T] \geq 2$ and $E[F] = \Theta(\lambda E[T])$. □

The proof above contains a fact that might be useful in other applications, so we formulate it as a separate lemma.

Lemma 11 *Let Ω be some set of size n. Let $k \in \mathbb{N}$. For all $i \in [k]$ let $s_i \leq n$ and S_i be a random subset of cardinality s_i of Ω. Let the S_i be stochastically independent. For $\omega \in \Omega$ let X_ω be the indicator random variable for the event that some S_i contains ω. Then the random variables X_ω, $\omega \in \Omega$, are negatively correlated.*

If in the lemma above we take $X_\omega^{(i)}$ as the indicator random variable of the event $\omega \in S_i$, then the $X_\omega^{(i)}$, $\omega \in \Omega$, are negatively correlated, see, e.g., the text following Theorem 1.16 in [11]. By definition, X is the point-wise maximum of the $X^{(i)}$, that is, we have $X_\omega := \max\{X_\omega^{(i)} \mid i \in [k]\}$ for all $\omega \in \Omega$. It would be a nice extension the lemma above to show that the point-wise maximum of arbitrary independent families of negatively correlated random variables is itself negatively correlated. We do not know if this statement is true.

We now turn to the case that k is small, say $\omega(1) \leq k \leq \sqrt{n}$, which is challenging in that here indeed the optimal expected runtime can show up. Consequently, there is no room for wasteful estimates. We regard the situation that also d is small, say between $n^{1/8}$ and $n^{1/4}$. In this case, the expected number of good bits in a mutation offspring is $O(n^{-1/4})$, consequently, with high probability the best mutation offspring only has at most a constant number of good bits. The expected number of bad bits in a mutation offspring is much larger, namely $\Theta(k) = \omega(1)$. Hence a progress only occurs if crossover selects at least one of the few good bits and selects fewer bad bits (out of the many that are present). We use this to carefully compute the drift and then apply the multiplicative drift theorem.

Lemma 12 *If $\omega(1) = k \leq \sqrt{n}$, then*

$$E[F] = \Omega\left(n \log n \min\{\tfrac{\exp(\Omega(r))}{\lambda r}, \tfrac{n^3}{\lambda}, \tfrac{\exp(\Omega(k))}{k}\}\right).$$

Proof We first analyze the progress made in an iteration starting with a search point with fitness distance between $n^{1/8}$ and $n^{1/4}$ and then use this information with the lower bound multiplicative drift theorem to obtain the claimed lower bound for the expected optimization time.

Consider an iteration starting with a search point x with $n^{1/8} \leq d(x) \leq n^{1/4}$. Let z be a search point among $\{x, x', y\}$ with maximal fitness. We aim at estimating the expected progress $E[d(x) - d(z)] = E[\textsc{Om}(z) - \textsc{Om}(x)]$. Since ℓ is binomially distributed, we have $\ell < k/2$ with probability at most $\exp(-\Omega(k))$ by the multiplicative Chernoff bound. Similarly, with probability at most $\exp(-\Omega(k))$, we have $\ell > 2k$. In this case, we have $E[\ell | \ell > 2k] \leq 3k+1$ by Lemma 1. Hence $E[\ell | \ell \notin [k/2, 2k]] = O(k)$.

Let $\tilde{x}$ be an offspring created in the mutation phase. Let $\tilde{g} := g(x, \tilde{x})$. Conditioning on the outcome of ℓ, $\tilde{g}$ has a hypergeometric distribution with parameters n, ℓ, and d. Hence $E[\tilde{g}] = d\ell/n$. For the mutation winner x', note that $g' := g(x, x') \leq \sum_{i=1}^{\lambda} g(x, x^{(i)})$. Hence $E[g'] \leq \lambda d\ell/n$.

For $\ell \notin [k/2, 2k]$, we use the estimate that $\textsc{Om}(z) - \textsc{Om}(x) \leq g'$ with probability one (note that this estimate is fulfilled both for $z = x'$ and $z = y$). Hence we compute

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell \notin [k/2, 2k]] \\
&\quad = \sum_{i \notin [k/2, 2k]} \Pr[\ell = i \mid \ell \notin [k/2, 2k]]\, E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell = i] \\
&\quad \leq \sum_{i \notin [k/2, 2k]} \Pr[\ell = i \mid \ell \notin [k/2, 2k]]\, E[g' \mid \ell = i] \\
&\quad \leq \sum_{i \notin [k/2, 2k]} \Pr[\ell = i \mid \ell \notin [k/2, 2k]]\, \lambda d i/n \\
&\quad = E[\ell \mid \ell \notin [k/2, 2k]]\, \lambda d/n = O(k\lambda d/n).
\end{aligned}$$

Hence let us assume (and condition on) that $k/2 \leq \ell \leq 2k$. Then $E[\tilde{g}] = d\ell/n \leq 2n^{-1/4}$ and thus $\Pr[\tilde{g} \geq 20] \leq O(n^{-5})$ by Theorem 1 (a)

and Theorem 3. Similarly, $E[g'] = \lambda d\ell/n \le 2\lambda dk/n$ and the probability that x' has a good bit at all is $\Pr[g' \ge 1] \le E[g'] = 2\lambda dk/n$ by Markov's inequality. If $g' = 0$, then $\textsc{Om}(x) = \textsc{Om}(z)$. So let us consider the case that $g' > 0$. Without conditioning on $g' > 0$, we have $\Pr[g' \ge 20] \le \lambda \Pr[\tilde{g} \ge 20] = O(\lambda n^{-5})$. Hence conditional on $g' > 0$, this probability is at most $O(\lambda n^{-5}/\Pr[g' \ge 1]) = O(\lambda n^{-5}/\min\{1, 2\lambda dk/n\}) = O(\max\{\lambda n^{-5}, n^{-4}/(dk)\})$. In this rare event, we can safely estimate $\textsc{Om}(z) - \textsc{Om}(x) \le n$, so let us turn to the more interesting case that $1 \le g' < 20$. Since $H(x, x') = \ell$, we have $b(x, x') \ge \ell - 19$. Consequently, $\ell = \Theta(k) = \omega(1)$ implies that no mutation offspring can be better than x. Let $\tilde{y}$ be an offspring generated in the crossover phase. Let $b_c := b(x, x', \tilde{y})$ denote the number of bad bits of x' that make it into $\tilde{y}$. For $\textsc{Om}(\tilde{y}) > \textsc{Om}(x)$ to hold, we need that $b_c \le 19$, but also that at least one good bit makes it into $\tilde{y}$, that is, $g(x, x', \tilde{y}) \ge 1$. Since b_c follows a binomial distribution with parameters $b(x, x')$ and r/k, we have $E[b_c] = b(x, x')r/k \ge (\ell - 19)r/k$. Hence $\Pr[b_c \le 19] \le \exp(-\Omega(r))$ by the multiplicative Chernoff bound. The expected number of good bits making it into $\tilde{y}$ is at most $E[g(x, x', \tilde{y})] \le 19 \cdot (r/k)$, hence by Markov's inequality this is also an upper bound for the probability that good bits make it into $\tilde{y}$ at all. Putting all this together and taking a union bound over the λ crossover offspring, we see that (still in the case that $1 \le g' \le 19$) the probability that some crossover offspring is better than x is at most $\lambda \cdot (19r/k) \cdot \exp(-\Omega(r))$; only then we have $\textsc{Om}(z) > \textsc{Om}(x)$, however, the gain is at most 19. Consequently,

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \le \ell \le 2k \wedge 1 \le g' \le 19] \\
&\quad \le \Pr[\textsc{Om}(z) > \textsc{Om}(x) \mid k/2 \le \ell \le 2k \wedge 1 \le g' \le 19] \cdot 19 \\
&\quad \le 19\lambda(19r/k)\exp(-\Omega(r))
\end{aligned}$$

We thus have

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \le \ell \le 2k] \\
&= \Pr[g' \ge 1] E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \le \ell \le 2k \wedge g' \ge 1] \\
&= (\lambda dk/n) \\
&\quad \big(\Pr[g' \le 19 \mid g' \ge 1] E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \le \ell \le 2k \wedge 1 \le g' \le 19] \\
&\quad + \Pr[g' \ge 20 \mid g' \ge 1] E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \le \ell \le 2k \wedge g' \ge 20]\big) \\
&\le (\lambda dk/n)\big(1 \cdot 19^2 \lambda r \exp(-\Omega(r))/k) + O(\max\{\lambda n^{-5}, n^{-4}/(dk)\})n\big) \\
&\le O(\lambda^2 dr \exp(-\Omega(r))n^{-1} + \lambda^2 dk n^{-5} + \lambda n^{-4}) \\
&= O(d\lambda^2(r\exp(-\Omega(r))n^{-1} + n^{-4})).
\end{aligned}$$

Together with the exceptional case that $\ell \notin [k/2, 2k]$, we obtain

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x)] \\
&\quad = \Pr[k/2 \le \ell \le 2k] E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \le \ell \le 2k] \\
&\qquad + \Pr[\ell \notin [k/2, 2k]] E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell \notin [k/2, 2k]] \\
&\quad = O\left(d\lambda^2(r\exp(-\Omega(r))n^{-1} + n^{-4})\right) + \exp(-\Omega(k))O(k\lambda d/n) \\
&\quad = O\left(\tfrac{d\lambda}{n}\left(\lambda r \exp(-\Omega(r)) + \lambda n^{-3} + k\exp(-\Omega(k))\right)\right).
\end{aligned}$$

We now use the lower bound multiplicative drift theorem (Theorem 5) to prove our claim. By Proposition 1, with high probability a run of the $(1 + (\lambda, \lambda))$ GA once encounters a search point x_0 with $d(x_0) \in [0.5n^{1/4}, n^{1/4}]$. For this case, we give a lower bound for the expected optimization time (which implies asymptotically the same bound for the general case). Denote by x_t, $t \geq 0$, the sequence of search points x generated by the $(1 + (\lambda, \lambda))$ GA in the sequel (except that x_t is the optimum solution from the first point on that the optimum was found). Let $s_{\min} := n^{1/8}$. We just showed that $E[d(x_{t+1}) - d(x_t) | d(x_t) = s] \leq s\delta$ holds for all $s \in [s_{\min}, d(x_0)]$, where we set $\delta = K\frac{\lambda}{n}(\lambda r \exp(-\Omega(r)) + \lambda n^{-3} + k\exp(-\Omega(k)))$ for some absolute constant K. By Proposition 1 again, we know that

$$\Pr[d(x_t) - d(x_{t+1}) \geq 0.5s \mid d(x_t) = s] \leq \lambda(\lambda + 1)\exp(-s) \leq 0.5\delta/\ln(s). \quad (7)$$

Consequently, we may apply Theorem 5 to the random process $(\max\{1, d(x_t)\})_{t \geq 0}$, and learn that the expected first t such that $d(x_t) \leq s_{\min}$ is

$$\Omega(\log(n)/\delta) = \Omega\left(\frac{n \log n}{\lambda(\lambda r \exp(-\Omega(r)) + \lambda n^{-3} + k\exp(-\Omega(k)))}\right).$$

Consequently, $E[T]$ is at least this number. By (7), we also have $E[T] \geq 2$ and thus

$$\begin{aligned} E[F] &= \Omega(\lambda E[T]) = \Omega\left(\frac{n \log n}{\lambda r \exp(-\Omega(r)) + \lambda n^{-3} + k\exp(-\Omega(k))}\right) \\ &= \Omega\left(n \log n \min\{\tfrac{\exp(\Omega(r))}{\lambda r}, \tfrac{n^3}{\lambda}, \tfrac{\exp(\Omega(k))}{k}\}\right). \end{aligned}$$

□

The following result exploits the simple fact that if in one iteration a mutation strength of ℓ was used, then regardless of the population size no progress of more than ℓ can be made.

Lemma 13 *Let $k \leq n/4$. Then $E[F] = \Omega(\frac{n\lambda}{k})$.*

Proof Let x_0 be the random initial search point. When x_t is defined for some $t \geq 0$, let x_{t+1} be the value of x after one iteration of the $(1 + (\lambda, \lambda))$ GA starting with $x = x_t$, unless this iteration generated the optimal solution, in this case let x_{t+1} be the optimal solution. Hence the sequence $(x_t)_t$ describes a typical run of the $(1 + (\lambda, \lambda))$ GA until the point when an optimal solution was generated. In particular, $T = \min\{t \geq 0 \mid d(x_t) = 0\}$.

We use the simple argument that all offspring generated in one iteration have a Hamming distance of at most ℓ from the parent. Consequently, $E[d(x_t) - d(x_{t+1})] \leq E[\ell] = k$, regardless of whether x_{t+1} is an optimal mutation offspring or the crossover winner. By the additive drift theorem (Theorem 4), we have $E[T|x_0] \geq d(x_0)/k$. Since the expected distance of a random search point from the optimum is $n/2$, the law of total expectation gives $E[T] \geq E[d(x_0)]/k = n/2k$. This is at least 2, so by Proposition 3, we have $E[F] = \Omega(\frac{n\lambda}{k})$. □

We now argue that for a wide range of values for the parameters and the fitness distance d, the expected progress per iteration is at most $O(r \log \lambda / \log\log \lambda)$. This immediately gives a lower bound of $\Omega(n \log\log \lambda/(r \log \lambda))$ via the additive drift theorem.

Lemma 14 *Let $k \leq n/80$, $\lambda \leq \exp(k/120)$, $\lambda = \exp(o(n))$, and $\lambda = \omega(1)$. Then $E[F] = \Omega(\frac{n\lambda \log\log(\lambda)}{r \log \lambda})$.*

Proof We shall show that the expected fitness gain in an iteration started with a search point with fitness distance at most $n/10$, is $O(r \log \lambda / \log\log \lambda)$. Since the (1+($\lambda$, λ)) GA by Proposition 1, here we use the assumption $\lambda = \exp(o(n))$, with high probability reaches once a search point x with $\textsc{Om}(x) \in [n/20, n/10]$, the claim follows from the additive drift theorem (Theorem 4).

To prove the drift condition, consider one iteration of the $(1 + (\lambda, \lambda))$ GA started with a parent individual x with $d(x) \leq n/10$. Let z be the value of x after one iteration, or the optimal search point if it was found as a mutation offspring (hence, as mutation winner). We show that the expected fitness gain $\textsc{Om}(z) - \textsc{Om}(x)$ is at most $O(\log \lambda / \log\log \lambda)$. For this, we first argue that we can assume that $k/2 \leq \ell \leq 2k$. Indeed, we have

$$\begin{aligned} E[\textsc{Om}(z) - \textsc{Om}(x)] = {} & \Pr[\ell < k/2]\, E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell < k/2] \\ & + \Pr[k/2 \leq \ell \leq 2k]\, E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \leq \ell \leq 2k] \\ & + \Pr[\ell > 2k]\, E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell > 2k]. \end{aligned}$$

By the multiplicative Chernoff bounds of Theorem 1, both $\Pr[\ell < k/2]$ and $\Pr[\ell > 2k]$ are $\exp(-\Omega(k))$. Since all offspring generated in one iteration (in either mutation and crossover phase) have Hamming distance at most ℓ from x, we immediately have $E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell < k/2] < k/2$. By Lemma 1, we also have $E[\textsc{Om}(z) - \textsc{Om}(x) \mid \ell > 2k] \leq E[\ell \mid \ell > 2k] \leq 3k + 1$. Hence

$$\begin{aligned} E[\textsc{Om}(z) - \textsc{Om}(x)] &\leq k \exp(-\Omega(k)) + E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \leq \ell \leq 2k] \\ &\leq O(1) + E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \leq \ell \leq 2k]. \end{aligned}$$

Hence we can assume for the remainder that $k/2 \leq \ell \leq 2k$. In this case, we argue as follows. Consider a mutation offspring $\tilde{x}$ and let $\tilde{g} := g(x, \tilde{x})$. Then $E[\tilde{g}] = \ell d(x)/n \leq \ell/10$. The probability that $\tilde{g} \geq \ell/5$ is at most $\exp(-(\ell/10)/3)) \leq \exp(-k/60)$ by Theorem 1 (b)[2] and Theorem 3. Since $\lambda \leq \exp(k/120)$, we see that with probability at least $1 - \exp(-k/120)$, all mutation offspring have at most $\ell/5$ good bits, implying that $g' := g(x, x')$ satisfies $g' \leq \ell/5$. Note that in the rare case that $g' > \ell/5$, which occurs with

[2] To be precise, we use here the fact that the bound of Theorem 1 (b) is also valid if both occurrences of $E[X]$ are replaced by an upper bound for $E[X]$. This is a well-known fact, but seemingly a reference is not so easy to find. Hence the easiest solution is maybe to derive this fact right from Theorem 1 (b) by extending the sequence $X_1, \ldots, X_n$ of random variables by random variables that take a certain value with probability one. By this, we can artificially increase $E[X]$ without changing the random variable $X - E[X]$. Hence the bound obtained from applying the Theorem to the extended sequence applies also to the original one.

probability at most $\exp(-k/120)$, we still have $\textsc{Om}(z) - \textsc{Om}(x) \leq g' \leq \ell \leq 2k$ with probability one, that is, this case contributes only another $k \exp(-\Omega(k))$ to the drift.

Therefore, let us now also condition on $g' \leq \ell/5$. Note that this also implies that $b' := b(x, x')$ satisfies $b' \geq (4/5)\ell$, since all mutation offspring have Hamming distance exactly ℓ from the parent x. Consequently, all mutation offspring are worse than x, and $z \in \{x, y\}$.

We now analyze the result of a crossover phase. Consider a crossover offspring $y^{(j)}$ and let $g_j := g(x, x', y^{(j)})$. Then $E[g_j] \leq g'r/k \leq (\ell/5) \cdot (r/k) \leq (2/5)r$. Let $\Delta = \frac{2r \ln(\lambda)}{\ln \ln(\lambda)} + s$ for a non-negative integer s. By Theorem 1 (a),

$$\begin{aligned}\Pr\Big[\max_{j \in [\lambda]} g_j \geq \Delta\Big] &\leq \sum_{j=1}^{\lambda} \Pr[g_j \geq \Delta] \leq \lambda \left(\frac{eE[g_j]}{\Delta}\right)^{\Delta} \\ &\leq \lambda \left(\frac{e \ln \ln(\lambda)}{5 \ln(\lambda)}\right)^{2\frac{\ln(\lambda)}{\ln \ln(\lambda)} + s} \leq 2^{-s}.\end{aligned}$$

Consequently, by Lemma 2,

$$E\Big[\max_j g_j\Big] = \sum_{t=1}^{\infty} \Pr\Big[\max_j g_j \geq t\Big] \leq \frac{2r \ln \lambda}{\ln \ln \lambda} + \sum_{s=1}^{\infty} 2^{-s} \leq \frac{2r \ln \lambda}{\ln \ln \lambda} + 1.$$

Clearly, the number of surviving good bits is an upper bound on the progress $\textsc{Om}(z) - \textsc{Om}(x)$. Hence the expected progress of one iteration, conditional on the assumptions made before, is at most

$$E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \leq \ell \leq 2k \wedge |G'| \leq \ell/5] \leq \tfrac{2r \ln \lambda}{\ln \ln \lambda} + 1.$$

Since the drift is always bounded by $\ell \leq 2k$, we have in fact

$$E[\textsc{Om}(z) - \textsc{Om}(x) \mid k/2 \leq \ell \leq 2k \wedge |G'| \leq \ell/5] \leq \min\{2k, \tfrac{2r \ln \lambda}{\ln \ln \lambda} + 1\}.$$

The unconditional drift thus is

$$\begin{aligned}E[\textsc{Om}(z) - \textsc{Om}(x)] &\leq \min\{2k, \tfrac{2r \ln \lambda}{\ln \ln \lambda} + 1\} + O(k) \exp(-\Omega(k)) \\ &= O(\min\{2k, \tfrac{2r \ln \lambda}{\ln \ln \lambda}\}).\end{aligned}$$

The additive drift theorem (Theorem 4), keeping in mind that we start with a search point with distance at least $n/20$, hence yields $E[T] = \Omega(\max\{(n/20)/2k, (n/20)\frac{\ln \ln \lambda}{2r \ln \lambda}\})$. This is at least 2, so we conclude $E[F] = \Omega(\lambda E[T]) \geq \Omega(n \frac{\lambda \ln \ln \lambda}{2r \ln \lambda})$. □

The above partial results already give that the best possible expected runtime of the $(1 + (\lambda, \lambda))$ GA among all parameter settings is $\Theta(F^*)$, and any such parameter set uses $\lambda = \Theta(\lambda^*)$ and $r = \Theta(1)$. We now also narrow down the range of values for k which may lead to an optimal expected runtime. The main argument resembles the proof of Lemma 9, namely that a large k lets all mutation offspring look similar and have many bad bits, so that the mutation

phase is wasteful and the crossover phase finds it hard to generate better-than-expected offspring. While the general idea of the proofs of the following result and Lemma 9 are similar, the details necessarily show some significant differences. Since we aim at a result for much smaller values of k, including the case of k smaller than any positive power of n, we need to regard a fitness regime with d much closer to n and with a much smaller ratio of upper and lower end. This requires several arguments to be made more precise, but also to exploit the information that we already have about optimal parameter sets.

Lemma 15 *Let $\lambda = \Theta(\lambda^*)$, $k = \exp(\omega(\sqrt{\log(n)\log\log\log(n)/\log\log(n)}))$, $k \leq n/2$, and $r = \Theta(1)$. Then the expected runtime of the $(1+(\lambda,\lambda))$ GA with these parameters is $\omega(F^*)$.*

Proof We first analyze the progress the $(1 + (\lambda, \lambda))$ GA makes in one iteration starting with a search point x having fitness distance $d := d(x) \in [3\ln\ln(n)n/k, n/3] =: [d_0, d_1]$. Let z denote the new parent individual after one iteration (which is either x or y), or the optimal solution in case one of the mutation offspring generated in this iteration was optimal. To use a lower bound drift theorem later, we prove an upper bound for $E[d(x) - d(z)]$.

We first convince ourselves that it is very unlikely that a mutation offspring is better than x. This will allow us to only regard the situation that $z \in \{x, y\}$. For a mutation offspring $\tilde{x}$ to be better than the parent x, more zero-bits have to flip than one-bits, that is, $\tilde{g} := g(x, \tilde{x}) > b(x, \tilde{x}) =: \tilde{b}$. By a simple domination argument, we see that this event is most likely for $d(x) = n/3$, so let us assume this for the moment. Then $E[\tilde{g}] = k/3$ and $E[\tilde{b}] = 2k/3$. We have $\Pr[\tilde{g} \geq k/2] = \exp(-\Omega(k))$ and $\Pr[\tilde{b} \leq k/2] = \exp(-\Omega(k))$. Consequently, $\Pr[\textsc{Om}(\tilde{x}) \geq \textsc{Om}(x)] \leq \exp(-\Omega(k))$. We thus compute

$$\begin{aligned}
&E[\textsc{Om}(z) - \textsc{Om}(x)] \\
&\quad \leq E\Big[\max_{\tilde{x}} \max\{0, \textsc{Om}(\tilde{x}) - \textsc{Om}(x)\}\Big] + E[\max\{0, \textsc{Om}(y) - \textsc{Om}(x)\}] \\
&\quad \leq \lambda n \exp(-\Omega(k)) + E[\max\{0, \textsc{Om}(y) - \textsc{Om}(x)\}] \\
&\quad = O(n^{-2}) + E[\max\{0, \textsc{Om}(y) - \textsc{Om}(x)\}].
\end{aligned}$$

We proceed by analyzing the quality of the crossover winner. Let $\tilde{x}$ be a mutation offspring and x' be the winning individual of the mutation phase. Let $\tilde{g} = g(x, \tilde{x})$ and $g' = g(x, x')$ be their numbers of good bits. We have $E[\tilde{g}] = dk/n$ and, by our lower bound on d, $\Pr[\tilde{g} \geq 2dk/n] \leq \exp(-(dk/n)/3) \leq 1/\ln(n)$. Consequently,

$$E[\max\{0, \tilde{g} - 2dk/n\}] \leq (1/\ln(n))E[\tilde{g} - 2dk/n \mid \tilde{g} \geq 2dk/n] = dk/n\ln(n)$$

by Lemma 1. We have

$$\begin{aligned}
g' \leq \max_{\tilde{x}} g(x, \tilde{x}) &= (2dk/n) + \max_{\tilde{x}}(g(x, \tilde{x}) - 2dk/n) \\
&\leq (2dk/n) + \sum_{\tilde{x}} \max\{0, g(x, \tilde{x}) - 2dk/n\}
\end{aligned}$$

and thus $E[g'] \leq 2dk/n + \sum_{\tilde{x}} E[\max\{0, g(x, \tilde{x}) - 2dk/n\}] = 2dk/n + dk\lambda/n \ln(n) = (2 + o(1))dk/n$, where all summations and maxima are taken over all mutation offspring.

Consider an offspring $\tilde{y}$ generated in the crossover phase. Let $\tilde{g}_{\tilde{y}} := g(x, x', \tilde{y})$. Then $E[\tilde{g}_{\tilde{y}}] \leq E[g'] \cdot (r/k) = O(d/n)$. Since the crossover winner y is chosen among the crossover offspring $\tilde{y}$ such that $\textsc{Om}(\tilde{y})$, and equivalently, $d(x) - d(\tilde{y})$, is maximal, we have $d(x) - d(y) \leq \sum_{\tilde{y}} \max\{d(x) - d(\tilde{y}), 0\}$, where $\tilde{y}$ runs over all λ crossover offspring. Consequently, $E[d(x) - d(y)] = O(\lambda \frac{d}{n})$ and hence also $E[\textsc{Om}(z) - \textsc{Om}(x)] = O(\lambda \frac{d}{n})$.

Building on this drift statement, we now use Witt's lower bound result for multiplicative drift (Theorem 5). Consider a run of the $(1 + (\lambda, \lambda))$ GA. For $t = 0, 1, \ldots$, denote by x_t the search point x at the beginning of the $(t + 1)$st iteration, except when the algorithm previously had generated the optimal solution, then let x_t be the optimal solution. With probability $1 - o(1)$, there is a t_0 such that $n/6 \leq d(x_{t_0}) \leq n/3$. We show that in this case, we have an expected optimization time as claimed, which implies that also the unconditioned expectation is of the same order of magnitude.

For $t = 0, 1, \ldots$ define $X_t = \max\{d(x_{t_0+t}), 1\}$. Observe that $X_{t+1} \leq X_t$ for all $t \geq 0$. Let $s_{\min} := d_0$. Then we have shown above that if $X_t = s > s_{\min}$, then $E[X_t - X_{t+1}] \leq K\lambda s/n$ for some absolute constant K. Hence the first condition of the drift theorem is fulfilled with $\delta = K\lambda/n$. From Proposition 1 we know that $\Pr[X_{t+1} \leq s/2] \leq \lambda^2 \exp(-\Omega(s)) = \exp(-\Omega(s))$. Hence for n (and thus also s) sufficiently large, also the second condition of the drift theorem is satisfied (with $\beta = 1/2$). We may thus apply the theorem and derive that the first t such that $X_t \leq s_{\min}$ satisfies

$$E[t] = \Omega\left(\frac{\ln(X_0/s_{\min})}{\delta}\right) = \Omega\left(\frac{n \log(k/\log\log n)}{\lambda}\right) = \Omega\left(\frac{n \log(k)}{\lambda}\right).$$

Note that this, naturally, is a lower bound on $E[T]$. Consequently, $E[F] = \Omega(\lambda E[T]) = \Omega(n \log k) = \omega(F^*)$. □

5.3 General Suggestions for the Use of the $(1 + (\lambda, \lambda))$ GA with Static Parameters

We have proven above that if some offspring population size λ, some mutation probability $p = k/n$, and some crossover bias $c = r/k$ lead to the asymptotically best expected runtime for the $(1 + (\lambda, \lambda))$ GA on OneMax, then $\lambda = \Theta(\lambda^*) = \Theta(\sqrt{\log(n) \log\log(n) / \log\log\log(n)})$, $k = \Omega(\lambda^*) \cap \exp(\omega(\sqrt{\log(n) \log\log\log(n) / \log\log(n)}))$, and $r = \Theta(1)$.

A closer inspection of the proofs allows (in a semi-rigorous manner) to extract some hints on the parameter choice also for optimization problems beyond the OneMax function class. The most clear one is that $r = \Theta(1)$, that is, $pc = \Theta(1/n)$, seems to be a good choice. This was argued intuitively in [16] based on the fact that such a choice results in that $\text{cross}_c(x, \text{mut}_p(x))$

has the same distribution as applying standard bit mutation to x with the standard choice of $1/n$ for the mutation probability. This intuitive argument is somewhat imprecise due to the fact that one iteration of the $(1+(\lambda,\lambda))$ GA contains two intermediate selection phases, so that neither does the winner of the mutation phase have a standard bit mutation distribution (with rate p), nor does the winner of the crossover phase have its bits taken independently from the mutation winner with probability c. Nevertheless, as the proofs of Lemma 9 and 12 (for a large range of parameter settings) show, in many situations a super-constant value for r leads with high probability to the event that the crossover offspring takes much more "bad" bits from the mutation winner than it takes good bits. Conversely, an r-value of $o(1)$ together with a not too small value for k lead to a probability of $1-\Theta(r)$ for the crossover offspring being equal to the parent x, making it useless.

For the choice of λ, as with all population-based algorithms, it is obvious that larger values of λ can only be beneficial if the positive effects of the large population outnumber the higher cost for a single iteration. From Lemma 12 we see that, again for broad ranges of the other parameters, we pay for a too small λ when making progress is difficult. A small value of λ decreases both the chance to find some good bits in the mutation phase and the chance that the good bits are copied into a crossover offspring. This quadratic price for a small λ is worth the multiplicative increase of the effort of one iteration. A similar lesson could be deduced from the fitness-dependent or the self-adjusting choice of λ in [16] and Section 6, which both again suggest a larger value for λ when being closer to the optimum, which in the OneMax landscape means that it is harder to find an improvement.

For the choice of the mutation probability $p=k/n$, the proof of Lemma 15 shows that a large k can lead to the effect that all mutation offspring look similar. In this case, the mutation phase does not gain from the large k-value, whereas in the crossover phase the crossover bias of $c=r/k$ makes it difficult to copy good bits into the final solution.

6 Self-Adjusting Parameter Choices

The linear expected optimization time obtained in [15] is a big success in the theory of evolutionary algorithms as it proves for the first time a super-constant speed-up via a fitness-dependent parameter choice. From the practical point of view, though, the question remains how in an actual application the user of the $(1+(\lambda,\lambda))$ GA would guess the fitness-dependent optimal choice of λ. In this section, we show that this is not needed. A self-adjusting choice inspired by the classic one-fifth rule can give the same (optimal, as the result in Section 6.5 shows) linear expected optimization time. To the best of our knowledge, this is the first result proving a reduced expected optimization time via parameter self-adjustment for an EA in discrete search spaces.

This section is organized as follows. We first provide some general background on dynamic parameter settings in Section 6.1. We also summarize

and extend in that section the classification scheme of Eiben, Hinterding, and Michalewicz [31] for parameter settings and discuss existing results for dynamic parameter choices in discrete optimization, with a focus on adaptive schemes. The self-adjusting $(1 + (\lambda, \lambda))$ GA and its inspiration from the one-fifth success rule are presented in Section 6.2. In Section 6.3 we present the runtime analysis for this algorithm, followed by a discussion of some general insights obtained through that analysis (Section 6.4). Finally, we discuss in Section 6.5 that the one-fifth success rule indeed suggests asymptotically optimal parameter settings.

6.1 Dynamic Parameter Settings—Overview, Terminology, and Results

It is widely acknowledged that setting the parameters of EAs is one of the key difficulties in evolutionary optimization. Eiben, Hinterding, and Michalewicz [31] call this challenge "one of the most important and promising areas of research in evolutionary computation". This statement retains its topicality 15 years after the original publication of [31] as many talks at evolutionary computation conferences certify. We also understand today that even small changes in the parameters can cause exponential performance gaps of the regarded algorithms [23, 29].

In the early seventies, substantial research efforts have been undertaken to find good parameter settings for general EAs, see for example [9]. Around the same time it has been discovered that it may be sub-optimal to use a fixed set of parameters throughout the whole optimization process. It was suggested instead to change the parameters of the algorithms by some dynamic update rules, often using some sort of feedback of the fitness landscape that the algorithm is facing. For example, it can be beneficial in earlier parts of the process to invest in *exploration* of the fitness landscape, while the algorithm should become more stable and focus on one or few areas of attraction in the later *exploitation* phase(s).

Since the literature is unanimous with respect to the terminology for parameter settings, we have adopted and slightly extended in this work the taxonomy of Eiben, Hinterding, and Michalewicz [31]. Figure 1, an adapted version of Figure 1 in [31], illustrates this classification, which we briefly summarize below.

The efforts of choosing the right parameters in an evolutionary algorithm is called *parameter setting*. The first difference is between *static* and *dynamic* parameter settings. In the former the parameters are set before the actual run of the algorithm and they are not changed during the optimization process. In typical applications, a *parameter tuning* step precedes the application of the EA. In this phase, suitable parameter choices are sought through initial experimental investigations, either for all parameters simultaneously, or in an iterative process.

Optimizing dynamic parameter choices is called *parameter control* in [31]. Three principals are discussed: deterministic, adaptive, and self-adaptive pa-

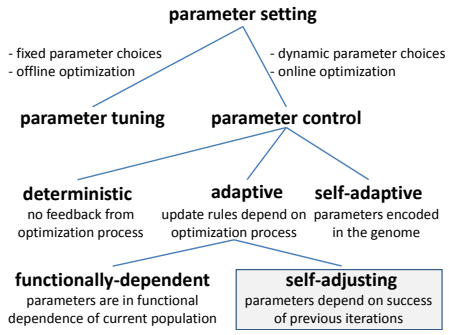

Fig. 1 An extended version of the classification scheme from [31]. We regard in this work self-adjusting parameter choices.

rameter control. A dynamic parameter choice is called *deterministic* if it does not depend on the fitness landscape encountered by the algorithm. That is, there is no feedback between the fitness values and the dynamic parameters. As noted in [31] this does not exclude randomized update schemes. A possibly better wording would therefore be *fixed* or *feedback-free* update rules. *Adaptive* parameter choices are those dynamic rules where the update rule depends on the optimization process. Within this class (and this is different from the classification scheme proposed in [31]) we distinguish between *functionally-dependent* parameter choices (where the parameters depend only on the current state of the algorithm, i.e., the current population) and *self-adjusting* adaptive choices, where the parameters depend on the success of (all) previous iterations. It is easy to see that the one-fifth success rule considered in this paper classifies as a self-adjusting parameter setting, while the fitness-dependent parameter choice considered in Theorem 10 is an example for a functionally-dependent parameter choice. Finally, *self-adaptive* parameter choices are encoded themselves in the genome of the search points and are subject to variation operators. The hope is that the better parameter choices yield better offspring and are thus more likely to survive the evolutionary process.

Interestingly, while in continuous domains parameter control mechanisms can be analyzed also theoretically [3, 37, 39], dynamic parameter choices play only a marginal role in theoretical investigations of EAs for discrete search spaces. The few exceptions showing an advantage of *adaptive* parameter settings in discrete optimization are (1) rank-based mutation rates for a $(\mu + 1)$ EA on function with a "trap" (guiding the search to local optima which are far from the global ones) and similarly artificial problem classes regarded in [51], (2) the fitness-dependent mutation rate for the (1+1) EA on LeadingOnes analyzed in [5], (3) a fitness-dependent mutation rate for the $(1 + \lambda)$ EA on OneMax analyzed in [4], and (4) the fitness-dependent choice of the population size in [16] for the $(1 + (\lambda, \lambda))$ GA on OneMax, respectively. All these

adaption schemes, however, require a very solid understanding of the problem at hand and are thus likely to be of limited practical relevance. No advantage could be shown for the fitness-dependent mutation rates in an immune algorithm (IA) regarded in [63] (cf. [64] for an investigation of the expected runtime of a $(\mu + 1)$ version of this IA). Another example from the discrete EA literature showing an advantage of adaptive parameter settings are reductions of the parallel runtime (but not the total expected optimization time) for several test functions when doubling the number of parallel instances in a parallel EA after each unsuccessful iteration [47].

Some more recent results proving a substantial advantage of self-adjusting parameter choices (not for the population though) are the analysis of an optimal self-adjusting choice of k-bit mutation for a variant of Randomized Local Search on OneMax presented in [19], and a provably optimal self-adjusting choice of the step size in multi-valued versions of OneMax in [17].

Works on dynamic but non-adaptive parameter update schemes include the *deterministic* choice of the mutation rate in an $(1+1)$ EA regarded in [43], the deterministic cooling schedule of the temperature determining the selection mechanism of Simulated Annealing regarded in [60] for the optimization of instances of the minimum spanning tree problem, and a *self-adaptive* choice of the mutation strength in a non-elitist population considered in [8].

6.2 The Self-Adjusting $(1 + (\lambda, \lambda))$ GA and Its Relationship to the One-Fifth Success Rule

One of the earliest adaptive update rules suggested in the evolutionary computation literature is the *one-fifth success rule*. It was independently discovered in [10, 54, 55] and today constitutes one of the best known and most widely applied techniques in parameter control. Several empirical results (cf. [32] and references therein) suggest that EAs using the one-fifth rule for adaptive parameter control are quite capable of finding optimal or close to optimal parameter settings. Since the parameters are updated without the intervention of the user, such update mechanisms are a very convenient way to minimize parameter tuning efforts. Furthermore, the one-fifth success rule does not require any problem-specific knowledge and is thus widely applicable.

Originally, the one-fifth rule was designed to control the step size of evolution strategies. In intuitive terms, it suggests that if the probability to create an offspring of better than current-best fitness is greater than $1/5$, then the step size should be increased to hopefully speed-up the exploration, while it should be decreased if the probability is lower than $1/5$ (in order to increase the chance of a success). Today, this rule has found applications much beyond the adaptation of the step size. Here in this work we use it for adjusting the offspring population size of a genetic algorithm. Without going into details, we note that many other adaptive update rules have been experimented with in the evolutionary computation literature, cf. [31], [32, Chapter 8], and [44] for surveys.

Algorithm 2: The self-adjusting $(1 + (\lambda, \lambda))$ GA with mutation probability p, crossover bias c, and update strength F.

1 **Initialization:** Sample $x \in \{0,1\}^n$ uniformly at random (u.a.r.);
2 Initialize $\lambda \leftarrow 1$;
3 **Optimization: for** $t = 1, 2, 3, \ldots$ **do**
4 **Mutation phase:**
5 Sample ℓ from $\mathcal{B}(n, p)$;
6 **for** $i = 1, \ldots, \lambda$ **do** $x^{(i)} \leftarrow \mathrm{mut}_\ell(x)$;
7 Choose $x' \in \{x^{(1)}, \ldots, x^{(\lambda)}\}$ with $f(x') = \max\{f(x^{(1)}), \ldots, f(x^{(\lambda)})\}$ u.a.r.;
8 **Crossover phase:**
9 **for** $i = 1, \ldots, \lambda$ **do** $y^{(i)} \leftarrow \mathrm{cross}_c(x, x')$;
10 Choose $y \in \{y^{(1)}, \ldots, y^{(\lambda)}\}$ with $f(y) = \max\{f(y^{(1)}), \ldots, f(y^{(\lambda)})\}$ u.a.r.;
11 **Selection and update step:**
12 **if** $f(y) > f(x)$ **then** $x \leftarrow y$; $\lambda \leftarrow \max\{\lambda/F, 1\}$;
13 **if** $f(y) = f(x)$ **then** $x \leftarrow y$; $\lambda \leftarrow \min\{\lambda F^{1/4}, n\}$;
14 **if** $f(y) < f(x)$ **then** $\lambda \leftarrow \min\{\lambda F^{1/4}, n\}$;

In discrete search spaces, naturally, things are very different. However, we can still come up with a natural variant of the one-fifth success rule. Note that in the $(1+(\lambda,\lambda))$ GA (with the suggested choices $p = \lambda/n$ and $c = 1/\lambda$ for the mutation rate and the crossover rate, respectively), increasing λ will increase the success probability of one iteration, however, at the price of an increased number of function evaluations, that is, higher runtime. Consequently, it makes sense to increase λ when the empirical success probability is low (to speed up the process of finding an improvement), but to reduce it when the success probability is large (to hopefully save computational effort).

The algorithm: Taking the implementation of the one-fifth success rule considered in [2] and originally proposed in [45] as example, the following self-adjusting version of the $(1 + (\lambda, \lambda))$ GA has been designed in [16], see also Algorithm 2. After an iteration that led to an increase of the fitness of x ("success"), indicating an easy success, we reduce λ by a constant factor $F > 1$ (of course, not letting λ drop below 1). If an iteration was not successful, we increase λ by a factor of $F^{1/4}$ (since we analyze the algorithm for mutation probability $p = \lambda/n$ we do not let λ exceed n). Consequently, after a series of iterations with an average success rate of $1/5$, we end up with the initial value of λ (unless the lower barrier of 1 was hit).

As a technical remark we note that where an integer is required in Algorithm 2 (e.g., lines 6 and 9) we round λ to its closest integer; i.e., instead of λ we regard $\lfloor \lambda \rfloor = \lambda - \{\lambda\}$ if the fractional part $\{\lambda\}$ of λ is less than $1/2$ and we regard $\lceil \lambda \rceil := \lfloor \lambda \rfloor + 1$ otherwise.

In a series of experimental evaluations it was shown in [16, Section 4] and [36] that this self-adjusting $(1 + (\lambda, \lambda))$ GA has a promising performance on a number of standard optimization problems.

6.3 Runtime Analysis for the Self-Adjusting Algorithm

We show that the self-adjusting $(1 + (\lambda, \lambda))$ GA (with standard parameters $p = \lambda/n$ and $c = 1/\lambda$) solves the generalized OneMax problem in linear time when the self-adjusting speed factor F is not too large. To the best of our knowledge, this is the first time that in a discrete search environment a self-adjusting parameter choice is shown to be superior to any static choice. Indeed the $(1 + (\lambda, \lambda))$ GA is the first proven example in discrete evolutionary algorithmics where a non-fitness-dependent parameter choice reduces the expected optimization time by more than a constant factor. The results that come closest to this are the mentioned results from [5, 47], which are either constant factor reductions of the expected runtime (in case of [5]) or reductions of the parallel expected runtime but not the total number of function evaluations (in case of [47]).

The proof of our result is rather technical. For this reason, we are only able to show a linear expected optimization time when F is smaller than a certain constant F^*, but we do not make this F^* precise. In general, making implicit constants precise is a difficult task in runtime analysis, and for many much simpler problems the implicit constants are not known. In the experiments conducted in [15], see in particular Figure 4 there, all values $F \in [1, 2]$ worked well (recall that in Auger's implementation [2] $F = 1.5$ was used). At the end of this section, we give some indication why F-values larger than 2.25, however, may lead to an exponential expected optimization time.

Our main result is the following.

Theorem 9 *The expected optimization time of the self-adjusting* $(1 + (\lambda, \lambda))$ *GA with parameters* $p = \lambda/n$ *and* $c = 1/\lambda$ *on every generalized* OneMax *function is* $O(n)$ *for any sufficiently small update strength* $F > 1$.

To prove Theorem 9, roughly speaking, we show that the population sizes λ suggested by the one-fifth success rule are usually not very far from the fitness-dependent choice $\lambda^* = \lceil \sqrt{n/(n - \textsc{Om}(x))} \rceil$ analyzed in [16, Theorem 8] (which is restated below as Theorem 10). To prove Theorem 10, one first shows that with the fitness-dependent choice of λ, each iteration of Algorithm 1 has a constant success probability. This shows that the average time spend on each fitness level is at most constant. A simple fitness-level argument then yields the claimed $O(n)$ bound.

Theorem 10 (Theorem 8 in [16]) *The expected runtime of the* $(1 + (\lambda, \lambda))$ *GA with* $p = \lambda/n$, $c = 1/\lambda$, *and fitness-dependent choice of* $\lambda^* := \lceil \sqrt{n/(n - \textsc{Om}(x))} \rceil$ *on every generalized* OneMax *function* Om *is linear in* n.

Intuitively, if λ happens to be much larger than λ^*, the success probability of the $(1 + (\lambda, \lambda))$ GA is so large that with reasonably large probability one of the next iterations is successful and, as a consequence, the value of λ is then adjusted to its previous value divided by F, thus approaching again λ^*. This observation has also been made experimentally in [16]. Figure 5 in [16] shows the close relationship between the self-adjusting population sizes and

the optimal fitness-dependent ones for a typical run of the $(1+(\lambda,\lambda))$ GA on OneMax.

A key argument in the proof of Theorem 9 is the following lemma, which shows that for large values of λ the success probability of the $(1+(\lambda,\lambda))$ GA is indeed reasonably large. This lemma can be seen as a generalization of Lemma 7.

Lemma 16 *Let $x \in \{0,1\}^n$. Let $\lambda \geq C_0 \lceil \sqrt{n/(n-\textsc{Om}(x))} \rceil$. Let $q = q(\lambda)$ be the probability that one iteration of Algorithm 1 (with parameters $p = \lambda/n$ and $c = 1/\lambda$) starting in x is successful. There exists a constant C such that for all $C_0 > C$ we have $q > 1/5$.*

Proof (of Lemma 16) We use the same notation as in the description of Algorithm 2. For readability purposes we again write λ even if an integer is required. Let L be the random variable sampled in Line 5 of Algorithm 2; i.e., L is the number of bits that are flipped to create the offspring in the mutation phase. For any fixed $\varepsilon > 0$ and for any λ, the success probability q of increasing the fitness by at least one is (by the law of total probability) at least

$$\begin{aligned} &\Pr[L \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]] \cdot \\ &\quad \min\{\Pr[\textsc{Om}(y) > \textsc{Om}(x) \mid L = \ell] \mid \ell \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]\}. \end{aligned} \tag{8}$$

By Lemmas 5 and 6 in [16] it holds for any ℓ that

$$\Pr[\textsc{Om}(y) > \textsc{Om}(x) \mid L = \ell] \geq \left(1 - \left(\tfrac{\textsc{Om}(x)}{n}\right)^{\lambda\ell}\right)\left(1 - \left(1 - \tfrac{1}{\lambda}(1-\tfrac{1}{\lambda})^{\ell}\right)^{\lambda}\right).$$

It thus suffices to bound from below

(i) $\min\left\{\left(1 - \left(1 - \tfrac{1}{\lambda}(1-\tfrac{1}{\lambda})^{\ell}\right)^{\lambda}\right) \mid \ell \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]\right\}$,

(ii) $\min\left\{1 - \left(\tfrac{\textsc{Om}(x)}{n}\right)^{\lambda\ell} \mid \ell \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]\right\}$,

(iii) $\Pr[L \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]]$.

Bounding (i): For any $\ell \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]$ it holds that

$$\begin{aligned} \left(1 - \tfrac{1}{\lambda}(1-\tfrac{1}{\lambda})^{\ell}\right)^{\lambda} &\leq \left(1 - \tfrac{1}{\lambda}(1-\tfrac{1}{\lambda})^{(1+\varepsilon)\lambda}\right)^{\lambda} \leq \left(1 - \tfrac{1}{\lambda}(1/4)^{1+\varepsilon}\right)^{\lambda} \\ &\leq \exp(-(1/4)^{1+\varepsilon}). \end{aligned}$$

For $\varepsilon < 1/25$ we can thus bound expression (i) from below by 0.21.

Bounding (ii): We set $d := n - \textsc{Om}(x)$ and obtain, for $\ell \in [(1-\varepsilon)\lambda, (1+\varepsilon)\lambda]$,

$$\left(\tfrac{\textsc{Om}(x)}{n}\right)^{\lambda\ell} \leq \left(\tfrac{\textsc{Om}(x)}{n}\right)^{(1-\varepsilon)\lambda^2} \leq \left(1 - \tfrac{d}{n}\right)^{(1-\varepsilon){C_0}^2 n/d} \leq (1/e)^{(1-\varepsilon){C_0}^2}.$$

This expression is at most $1/100$ for large enough C_0, showing that we can bound (ii) from below by 0.99.

Bounding (iii): We apply Chernoff's bound to bound (iii) from below by $1 - 2\exp(-\varepsilon^2\lambda/3)$. Since $\lambda \geq 2C_0$, this term is again larger than 0.99 for a suitably chosen C_0.

Putting everything together we have seen that, for a suitable choice of C_0, the expression in (8) is strictly larger than $0.99^2 \cdot 0.21 > 1/5$. □

While the proof of Lemma 16 was rather straightforward, the proof of the main theorem, i.e., Theorem 9, is much more involved.

Proof (of Theorem 9) As in the overview given before Lemma 16 we sloppily denote in the following by λ^* our fitness-dependent parameter choice of Theorem 10; i.e., $\lambda^* := \lceil \sqrt{n/(n - \textsc{Om}(x))} \rceil$. Note that the value of λ^* depends on the current fitness value but that this is not reflected in the abbreviation. To increase the readability, we omit again to specify whether λ has to be rounded up or down.

We partition the optimization process into phases. The first phase starts with the first fitness evaluation. A phase ends with an iteration at whose end we have increased the fitness of x and $\lambda \leq C_0\lambda^*$ holds, for a sufficiently large constant C_0 that we do not compute explicitly. (C_0 is determined by Lemma 16.)

We shall first show that each phase has an expected cost of $O(\lambda^*)$ fitness evaluations. From this is it not difficult to conclude the proof by arguments used in the proof of Theorem 10.

To bound the expected cost of each phase, we distinguish between *"short"* phases, in which $\lambda < C_0\lambda^*$ holds throughout, and *"long"* phases, in which $\lambda \geq C_0\lambda^*$ for at least one iteration. We abbreviate the threshold $C_0\lambda^*$ by $\bar{\lambda}$. Note that $\bar{\lambda}$ as well depends on the current fitness $\textsc{Om}(x)$.

Claim 1: The expected cost of a short phase is $O(\lambda^*)$.

Proof of claim 1: Let $\tilde{\lambda}$ be the value of λ at the beginning of the phase and let t be the number of iterations of the phase. Since λ does not exceed the threshold $\bar{\lambda}$, there is exactly one iteration in which the fitness of x is increased. That is, the value of λ has first been multiplied $t-1$ times by $F^{1/4}$ until there was a fitness increase and the λ-value has been shrunk as a consequence of the fitness increase. The value of λ at the end of the phase is thus $\tilde{\lambda}F^{(t-1)/4-1}$ and this value is bounded from above by $\bar{\lambda}$ (since we are considering a short phase). Since an iteration with parameter λ requires 2λ fitness evaluations, the total number of fitness evaluations is thus

$$\sum_{i=0}^{t-1} 2\tilde{\lambda}F^{i/4} = 2\tilde{\lambda}\frac{F^{t/4} - 1}{F^{1/4} - 1} = O(\bar{\lambda}) = O(\lambda^*). \tag{9}$$

Claim 2: The expected cost of a long phase is $O(\lambda^*)$.

Proof of claim 2: We split the long phase into an opening phase and a main phase. The *opening phase* ends with the last iteration in which $\lambda < \bar{\lambda}$ holds, so that the *main phase* starts with a λ that is at least as large as $\bar{\lambda}$, but less than $\bar{\lambda}F^{1/4}$.

Let T denote the number of fitness evaluations during the phase and let I denote the number of iterations in the main (!) phase. As in the proof of Claim 1 it will be easy to see that $\mathrm{E}[T \mid I = t] = D\lambda^* F^{t/4}$ for all t and some fix constant D. The most technical part of this proof is to bound the probability that the main phase of a long phase requires t iterations; i.e., $\Pr[I = t]$ given that we are in a long phase. We show that this probability is at most $\exp(-ct)$ for some positive constant c. It is well known that the geometric series $(\sum_{t=1}^{\ell} F^{t/4}\exp(-ct))_{\ell\in\mathbb{Z}_{>0}}$ converges if $F^{1/4} < \exp(c)$. The overall expected number of fitness evaluations during a long phase is thus

$$\sum_{t=1}^{\infty} \mathrm{E}[T \mid I = t]\Pr[I = t] \leq D\lambda^* \sum_{t=1}^{\infty} F^{t/4}\exp(-ct),$$

which is $O(\lambda^*)$ as desired.

It remains to prove the following two claims.

Claim 2.1: $\mathrm{E}[T \mid I = t] \leq D\lambda^* F^{t/4}$ for some large enough constant D.

Claim 2.2: Given that we are in a long phase, we have $\Pr[I = t] = \exp(-ct)$ for a positive constant c.

Proof of Claim 2.1: The cost of the opening phase is at most $2\tilde{\lambda}\sum_{i=0}^{m} F^{i/4}$, where $\tilde{\lambda}$ is the initial value of λ at the beginning of the phase and m is chosen maximally such that $\lambda F^{m/4} < \bar{\lambda}$. As in (9) one shows that this sum is $O(\lambda^*)$. Similarly, since the initial λ of the main phase is at most $\bar{\lambda}F^{1/4}$, the cost of the main phase is at most

$$\bar{\lambda}\sum_{i=1}^{t} F^{i/4} = \bar{\lambda}(F^{(t+1)/4} - 1)/(F^{1/4} - 1) \leq D'\bar{\lambda}F^{(t+1)/4}$$

for $D' \geq 1/(F^{1/4} - 1)$.

Proof of Claim 2.2: As mentioned above, the main phase starts with a $\tilde{\lambda}$ that is at least $\bar{\lambda}$ and strictly less than $\bar{\lambda}F^{1/4}$. We are interested in the first point in time at which λ is less than $\bar{\lambda}$. Note that all future values encountered in this phase are of the type $\tilde{\lambda}F^r$ for r being a multiple of $1/4$. By regarding this exponent r, we transform the process into a biased random walk on the line $(1/4)\mathbb{Z}$. Our starting position is 0. If an iteration is successful, i.e., if the fitness value of x has increased during the iteration, the process does one step of length one to the left. It does a step of length $1/4$ to the right otherwise (we thus pessimistically ignore the fact that λ never exceeds n). We bound the probability that it takes t or more iterations until this random walk has reached a value of less than 0. At this point in time the current λ is less than the original $\bar{\lambda}$ value that was active at the beginning of the main phase. That is, when the random walk reaches a position smaller than 0, λ is for certain less than the then active threshold $\bar{\lambda}$ (which, by definition, increases whenever the fitness value of x does).

If an iteration is successful with probability at least q, the expected progress of this random walk in one iteration is $(1 - q)/4 - q$, which is negative since

by Lemma 16 we have $q > 1/5$. Hence there exists a constant $c > 0$ such that $(1 - q)/4 - q < -c$.

To conclude the proof of Claim 2.2, let us define random variables X_i, $1 \leq i \leq t$, by setting $X_i = 1/4$ if the fitness does not increase in the ith iteration of the main phase, and setting $X_i = -1$ otherwise. We have just seen that $\mathrm{E}[X_i] \leq -c$. Given that we are in a long phase, the probability that the main phase has length at least t equals $\Pr[\forall j < t : \sum_{i=1}^{j} X_i > 0]$. This is at most $\Pr[\sum_{i=1}^{t-1} X_i > 0]$, which is in turn bounded from above by $\Pr[\sum_{i=1}^{t-1} X_i > \mathrm{E}[\sum_{i=1}^{t-1} X_i] + (t-1)c]$. We apply Chernoff's bound—confer Theorem 1.11 in [11] for a version that allows for random variables that do not necessarily take positive values—to see that, as desired, this term is at most

$$\exp\left(-\frac{2(t-1)^2c^2}{(t-1)(5/4)^2}\right) = \exp\left(-32(t-1)c^2/25\right).$$

□

6.4 General Insights from the Runtime Analysis

The analysis above reveals the following facts, which might be helpful in general when trying to use a one-fifth success rule or a related self-adjusting rule in discrete search spaces.

The adjustment rule must fit to the limiting success probability. In the proof above, it was crucial that the success probability shown in Lemma 16 was a constant larger than 1/5. It is easy to see that if the success probability was uniformly bounded from above by a constant $\sigma < 1/5$, then $\log_F(\lambda)$ in expectation would increase by a positive constant in each iteration. Consequently, λ would show an exponential growth, quickly leading to wastefully large values. This can partially be overcome by imposing an upper barrier for λ (we have such a barrier, namely $\lambda \leq n$, to ensure that the mutation probability $p = \lambda/n$ is at most 1), however, this would still lead to the algorithm mostly working with this maximal value of λ instead of a value close to the ideal λ^*.

In general, there is no reason for not trying success rules with other ratios $1/r$ than one-fifth, that is, increasing λ by $F^{1/(r-1)}$ instead of $F^{1/4}$ in case of non-success. In general, a larger value of r will slightly decrease the speed of adjustment, but is more likely to overcome the problem described in the previous paragraph. Note that for our problem, when $\lambda = \omega(1)$, the success probability is uniformly bounded from above by $(1 + o(1))e^{-1/e} \approx 0.31$. Consequently, the one-fifth rule avoids the exponential growth of λ, whereas a one-third rule or a one-half rule (e.g., doubling or halving the parameter as in [47]) would not.

The constant F matters. Even when the combination of update rule and success probability avoids an expected exponential growth of λ, things

can still go wrong when the update strength F is too large. Here is an example (where, to ease the presentation, we assume that we have no upper barrier on λ; with an upper barrier, as above, the problem remains, though possibly to a smaller extent): Imagine that we start with some value λ_0. Above, we saw that the success probability of an iteration is bounded from above by 0.31. Consequently, the probability of having exactly m consecutive non-successes is at least $0.31 \cdot 0.69^m$. The optimization time of the last iteration alone is $\lambda_0 F^{m/4}$. Consequently, the expected effort of finding one improvement is at least $0.31\lambda_0 \sum_{m=0}^{\infty}(0.69F^{1/4})^m$. When $0.69F^{1/4} > 1$, this series does not converge, i.e., the expected effort for one improvement is infinite. In our case, this happens (at least) when $F > 4.41$. Note that this was a rough estimate aimed at quickly demonstrating that large F-values can be dangerous. Better values can be achieved with more effort. E.g., the probability that among $6m$ iterations, we have at most m successes, is more than $(0.755 + o(1))^m$; this can be seen from approximating the binomial distribution with a normal distribution. Since this event also increases the initial λ value by $F^{m/4}$, the corresponding series already diverges for $F \geq 3.08$. Optimizing the ratio of successes and non-successes, we see that the probability of having γ successes among $1 + 5\gamma$ trials is more than $(0.8167 + o(1))^m$, showing that any $F \geq 2.25$ leads to an exponential expected optimization time. We do not know to what extent this argument can be improved. For this reason, we would rather suggest to choose a small value of F, clearly below 2, and trade in the possibly faster adjustment to the ideal parameter value for a reduced risk of an expected infinite optimization time.

6.5 A Linear Lower Bound for All Dynamic Parameter Choices

In the previous section we have seen that the expected optimization time of the self-adjusting $(1+(\lambda,\lambda))$ GA is better than that of the $(1+(\lambda,\lambda))$ GA with any static population size. We next show that the simple update rule is even asymptotically best possible among all dynamic parameter choices. That is, we prove that, regardless of how the parameters are updated in each iteration, the $(1 + (\lambda, \lambda))$ GA always has an expected runtime on OneMax that is at least linear in n.

To this aim, we consider the class of $(1+(\lambda,\lambda))$ GA versions with dynamic parameter choices following the scheme of Algorithm 3. It allows an arbitrary setting of the parameters depending on the whole history. We show that any such algorithm has an expected performance for the OneMax problem that is at least linear in n. Consequently, our self-adjusting $(1 + (\lambda, \lambda))$ GA is asymptotically optimal among this broad class of algorithms, which includes all fitness-dependent and all success-based parameter settings.

Theorem 11 *For any $(1 + (\lambda, \lambda))$ GA with dynamic parameter choices, that is, an instance of Algorithm 3, the optimization time F on* OneMax *satisfies* $\mathrm{E}[F] = \Omega(n)$.

Algorithm 3: A general framework for the $(1+(\lambda, \lambda))$ GA with dynamic parameter choices, maximizing a given function $f : \{0,1\}^n \to \mathbb{R}$. In each iteration, offspring population size λ, mutation probability p, and crossover bias c are chosen depending on the whole history $\mathcal{H}$ of the search process. The subscript $\mathcal{H}$ indicates this (arbitrary) dependence on the history.

Initialization: Choose $x \in \{0,1\}^n$ uniformly at random (u.a.r.);
Optimization: **for** $t = 1, 2, 3, \ldots$ **do**
  Mutation phase:
    Determine $\lambda_{\mathcal{H}} \in \mathbb{N}$, $p_{\mathcal{H}} \in [0,1]$, and $c_{\mathcal{H}} \in [0,1]$ depending on the whole history $\mathcal{H}$;
    Sample ℓ from $\mathcal{B}(n, p_{\mathcal{H}})$;
    for $i = 1, \ldots, \lambda_{\mathcal{H}}$ **do** $x^{(i)} \leftarrow \mathrm{mut}_\ell(x)$;
    Choose $x' \in \{x^{(1)}, \ldots, x^{(\lambda_{\mathcal{H}})}\}$ with $f(x') = \max\{f(x^{(1)}), \ldots, f(x^{(\lambda_{\mathcal{H}})})\}$ u.a.r.;
  Crossover phase:
    for $i = 1, \ldots, \lambda_{\mathcal{H}}$ **do** $y^{(i)} \leftarrow \mathrm{cross}_{c_{\mathcal{H}}}(x, x')$;
    Choose $y \in \{y^{(1)}, \ldots, y^{(\lambda_{\mathcal{H}})}\}$ with $f(y) = \max\{f(y^{(1)}), \ldots, f(y^{(\lambda_{\mathcal{H}})})\}$ u.a.r.;
  Selection step: **if** $f(y) \geq f(x)$ **then** $x \leftarrow y$;

Compared to our lower bound of Section 5, we claim here a weaker lower bound for a broader class of algorithms. Fortunately, this allows for a significantly easier proof. The main argument is that, independent of the parameters chosen by the algorithm, an iteration using an offspring population size of λ and starting with an individual of fitness at least $0.9n$ increases the fitness by at most $O(\lambda)$.

As a side remark, we note that the unrestricted black-box complexity of OneMax (or, more precisely, the unrestricted black-box complexity of the class $\{f_z \mid z \in \{0,1\}^n\}$ of generalized OneMax functions $f_z : \{0,1\}^n \to [0..n], x \mapsto |\{i \in [n] \mid x_i = z_i\}|$) is $\Theta(n/\log n)$, cf. [1,30,33]. That is, no black-box algorithm exists which can find the optimum for any of these functions in expected time $o(n/\log n)$, while there is a black-box algorithm that needs only about $2n/\log n$ function evaluations to determine the unique global optimum z of the (unknown) function f_z. On the other hand, the best known *binary unbiased* black-box algorithm for OneMax has linear runtime [24]. Intuitively speaking, cf. also our discussion in Section 3.1, a binary unbiased black-box algorithm is one that samples all search points from distributions that depend only on at most two previously queried ones and only from distributions that do not discriminate between the bit positions $1, \ldots, n$ nor between the bit values 0 and 1. The $(1+(\lambda, \lambda))$ GA is such a binary unbiased black-box algorithm. Whether its $O(n)$ runtime bound is best possible among all binary unbiased black-box algorithms or whether $o(n)$ binary unbiased black-box algorithms exits is a big open problem in black-box complexity.

Proof Let us fix a $(1 + (\lambda, \lambda))$ GA with dynamic parameter choices. We first show that the expected fitness gain in an iteration started with a search point

with fitness distance at most $n/10$ is at most $O(\lambda_{\mathcal{H}})$. To prove this statement, let $x \in \{0,1\}^n$ with $d(x) \leq n/10$. Consider an iteration of this $(1+(\lambda,\lambda))$ GA started with this search point as parent individual. Let λ, p, and c be the values of offspring population size, mutation probability, and crossover bias chosen by the algorithm depending on the previous search history. Denote by z the search point that forms the new parent individual after this iteration. Our aim is to show that $\mathrm{E}[f(z) - f(x)] = O(\lambda)$.

Let ℓ be the mutation strength sampled by the algorithm. We have $\ell \sim \mathcal{B}(n,p)$, but we shall not exploit this. Since $f(z) - f(x) \leq \ell$ with probability one, we can assume that $\ell \geq \lambda$ as otherwise we trivially have $f(z) - f(x) = O(\lambda)$.

We use in the following the notation introduced at the beginning of Section 5.2. The multiplicative Chernoff bound (see Theorem 1) and a union bound show that with probability at most $\lambda \exp(-\Omega(\ell)) \leq \ell \exp(-\Omega(\ell))$, there is a mutation offspring $\tilde{x}$ with $g(x,\tilde{x}) \geq \ell/5$. In this case, we simply estimate $f(z) - f(x) \leq \ell$ as before.

Otherwise, the mutation winner x' satisfies $g(x,x') \leq \ell/5$ and, consequently, $b(x,x') \geq (4/5)\ell$. Consider a crossover offspring $\tilde{y}$. Let A be the event that $g'(x,x',\tilde{y}) \geq (2/5)c\ell$ or $b'(x,x',\tilde{y}) \leq (2/5)c\ell$. Note that $f(\tilde{y}) \leq f(x)$ when A does not hold. By (i) this observation, (ii) the multiplicative Chernoff bound, (iii) the fact that crossover takes bits independently from x and x', and (iv) Lemma 1, we compute

$$\begin{aligned}
\mathrm{E}[\max\{0, f(\tilde{y}) - f(x)\}] &= \Pr[A]\,\mathrm{E}[\max\{0, f(\tilde{y}) - f(x)\} \mid A] \\
&\leq \exp(-\Omega(c\ell))\,\mathrm{E}[g'(x,x',\tilde{y}) \mid g'(x,x',\tilde{y}) \geq (2/5)c\ell] \\
&\leq \exp(-\Omega(c\ell))((2/5)c\ell + c\ell/5),
\end{aligned}$$

which is $O(1)$ regardless of the outcome of ℓ and the dynamic choice of c. For the crossover winner y, we thus have

$$\mathrm{E}[f(y) - f(x)] \leq \sum_{\tilde{y}} \mathrm{E}[\max\{0, f(\tilde{y}) - f(x)\}] = O(\lambda).$$

Taking the two cases together, we obtain $\mathrm{E}[f(z) - f(x)] = \exp(-\Omega(\ell))\ell^2 + O(\lambda) = O(\lambda)$ independent of the outcome of ℓ.

Summarizing the above, we have proven that there is a constant $C > 0$ such that in each iteration starting with a search point with fitness distance at most $n/10$ and using an offspring population size of λ, the expected fitness gain is at most $C\lambda$.

Consider now a run of the dynamic $(1 + (\lambda,\lambda))$ GA. To avoid technicalities, let us assume that the $(1 + (\lambda,\lambda))$ GA after having reached an optimal parent individual continues to run and does so using $\lambda = 1$ in each subsequent iteration. Considering such an infinite run, let t_1 be the first iteration after which $B := \frac{1}{80C}n$ fitness evaluations have been performed. Denoting by $\lambda_t, t \in \mathbb{N}$, the value of λ used in iteration t, we thus have $1 + \sum_{i=1}^{t_1} 2\lambda_i \geq B$ and $1 + \sum_{i=1}^{t_1-1} 2\lambda_i < B$.

Denote by x_t, $t \in \mathbb{N}$, the search point forming the parent population after iteration t. With probability $1 - \exp(-\Omega(n))$, the initial search point x_0 has a fitness between $0.4n$ and $0.6n$. We condition on this event in the following. We first show that with probability $1/2 - o(1)$, we have $d(x_{t_1-1}) \geq n/40$. Let $t_0 \in \mathbb{N}$ be minimal such that $d(x_{t_0}) \leq n/10$. If $t_0 > t_1 - 1$, there is nothing to show. Hence let us assume that $t_0 \leq t_1 - 1$. By Proposition 1, with probability at least $1 - B^2 \exp(-\Omega(n))$ we have $d(x_{t_0}) \geq n/20$. We condition again on this. By our above argument, we have

$$\mathrm{E}[f(x_{t_1-1}) - f(x_{t_0})] \leq C \sum_{i=t_0+1}^{t_1-1} \lambda_i \leq CB \leq \tfrac{1}{80} n.$$

By Markov's inequality, with probability at least $1/2$, we have $f(x_{t_1-1}) - f(x_{t_0}) \leq \frac{1}{40} n$, implying $d(x_{t_1-1}) \geq n/40$.

Hence with probability at least $1/2 - o(1)$, we have $d(x_{t_1-1}) \geq n/40$. Let us condition on this. Then all parent individuals up to iteration $t_1 - 1$ have a fitness distance of at least $\frac{1}{40} n$. By Proposition 1, with probability at least $1 - B \exp(-\Omega(n))$, all mutation offspring created up to iteration $t_1 - 1$ have a fitness distance of at most $n/80$. Consequently, up to the end of iteration $t_1 - 1$, no optimal solution has been found and $d(x_{t_1-1}) \geq n/40$.

If $\lambda_{t_1} \leq n$, then again by Proposition 1, the t_1-th iteration also with high probability creates no search point with fitness distance better than $n/80$. In this case, the optimization time F satisfies $F \geq B = \frac{1}{80C} n$. If $\lambda_{t_1} \geq n$, then Proposition 2 shows that the first n mutation offspring all have a fitness distance of at least $n/80$ (with high probability). In this case, trivially, $\mathrm{E}[F] \geq n$ as claimed.

Putting everything together, we see that with probability $1/2 - o(1)$, we have $F \geq \min\{B, n\} = \Omega(n)$, which gives the claim. □

7 Summary

In this work, we have conducted a detailed analysis of the optimization time of the $(1 + (\lambda, \lambda))$ GA on OneMax. In a first step, we have proven an improved upper bound for the expected optimization time and a tail bound for the upper tail of the runtime distribution. We also proved that this bound is tight. Even more, we showed that no parameter combination for the $(1 + (\lambda, \lambda))$ GA can lead to an asymptotically better expected runtime on the OneMax test function class than the one suggested in [16], where this algorithm was first proposed. We also proved that if some offspring population size λ, some mutation probability $p = k/n$, and some crossover bias $c = r/k$ leads to the asymptotically best expected runtime, then $\lambda = \Theta(\lambda^*) = \Theta(\sqrt{\log(n) \log\log(n) / \log\log\log(n)})$, $k = \Omega(\lambda^*) \cap \exp(\omega(\sqrt{\log(n) \log\log\log(n) / \log\log(n)}))$, and $r = \Theta(1)$. These analyses both show that the $(1 + (\lambda, \lambda))$ GA is faster than what could be proven in [16], and it suggests a slightly different value for the optimal offspring population size λ,

again leading to a super-constant factor speed-up over the runtime stemming from the value giving the best bound in the previous work.

Finally, we have analyzed the $(1+(\lambda,\lambda))$ GA with self-adjusting population sizes. We have shown that it optimizes any generalized OneMax function in linear time. This is best possible for any (static or dynamic) parameter choice and is better by a $\Omega(\sqrt{\log(n)\log\log\log(n)/\log\log(n)})$ factor than any $(1+(\lambda,\lambda))$ GA with static population size. Our result thus shows for the first time that self-adjusting parameter choices can be provably beneficial in discrete optimization problems.

We hope that our insights make it easier to use the $(1+(\lambda,\lambda))$ GA, which both in theoretical and empirical investigations showed a promising performance. We are also optimistic that the proof ideas developed in this work make future analyses of multi-dimensional parameter spaces easier. Furthermore, we hope that our results inspire more work on the runtime of evolutionary algorithms with self-adjusting parameter choices.

Acknowledgements 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 Programme Gaspard Monge en Optimisation et Recherche Opérationnelle. It is also supported by a second project within the "FMJH Program Gaspard Monge in optimization and operation research" program, and from the support to this program from EDF (Électricité de France).

References

1. Anil, G., Wiegand, R.P.: Black-box search by elimination of fitness functions. In: Proc. of Foundations of Genetic Algorithms (FOGA'09), pp. 67–78. ACM (2009)
2. Auger, A.: Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'09), (Companion), pp. 2447–2452. ACM (2009)
3. Auger, A., Hansen, N.: Linear convergence on positively homogeneous functions of a comparison based step-size adaptive randomized search: the (1+1) ES with generalized one-fifth success rule (2013). CoRR abs/1310.8397
4. Badkobeh, G., Lehre, P.K., Sudholt, D.: Unbiased black-box complexity of parallel search. In: Proc. of Parallel Problem Solving from Nature (PPSN'14), *Lecture Notes in Computer Science*, vol. 8672, pp. 892–901. Springer (2014)
5. Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In: Proc. of Parallel Problem Solving from Nature (PPSN'10), *Lecture Notes in Computer Science*, vol. 6238, pp. 1–10. Springer (2010)
6. Dang, D., Friedrich, T., Kötzing, T., Krejca, M.S., Lehre, P.K., Oliveto, P.S., Sudholt, D., Sutton, A.M.: Emergence of diversity and its benefits for crossover in genetic algorithms. In: Proc. of Parallel Problem Solving from Nature (PPSN'16), *Lecture Notes in Computer Science*, vol. 9921, pp. 890–900. Springer (2016)
7. Dang, D., Friedrich, T., Kötzing, T., Krejca, M.S., Lehre, P.K., Oliveto, P.S., Sudholt, D., Sutton, A.M.: Escaping local optima with diversity mechanisms and crossover. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'16), pp. 645–652. ACM (2016)
8. Dang, D.C., Lehre, P.K.: Self-adaptation of mutation rates in non-elitist populations. In: Proc. of Parallel Problem Solving from Nature (PPSN'16), *Lecture Notes in Computer Science*, vol. 9921, pp. 803–813. Springer (2016)
9. De Jong, K.A.: An analysis of the behavior of a class of genetic adaptive systems. Ph.D. thesis, Ann Arbor, MI, USA (1975). PhD Thesis

10. Devroye, L.: The compound random search. Ph.D. dissertation, Purdue Univ., West Lafayette, IN (1972)
11. Doerr, B.: Analyzing randomized search heuristics: Tools from probability theory. In: A. Auger, B. Doerr (eds.) Theory of Randomized Search Heuristics, pp. 1–20. World Scientific Publishing (2011). Available at http://www.worldscientific.com/doi/suppl/10.1142/7438/suppl_file/7438_chap01.pdf
12. Doerr, B.: Optimal parameter settings for the $(1+(\lambda, \lambda))$ genetic algorithm. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'16), pp. 1107–1114. ACM (2016)
13. Doerr, B., Doerr, C.: Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'15), pp. 1335–1342. ACM (2015)
14. Doerr, B., Doerr, C.: A tight runtime analysis of the $(1+(\lambda, \lambda))$ genetic algorithm on OneMax. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'15), pp. 1423–1430. ACM (2015)
15. Doerr, B., Doerr, C., Ebel, F.: Lessons from the black-box: Fast crossover-based genetic algorithms. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'13), pp. 781–788. ACM (2013)
16. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theoretical Computer Science **567**, 87–104 (2015)
17. Doerr, B., Doerr, C., Kötzing, T.: Provably optimal self-adjusting step sizes for multi-valued decision variables. In: Proc. of Parallel Problem Solving from Nature (PPSN'16), *Lecture Notes in Computer Science*, vol. 9921, pp. 782–791. Springer (2016)
18. Doerr, B., Doerr, C., Kötzing, T.: The right mutation strength for multi-valued decision variables. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'16), pp. 1115–1122. ACM (2016). Full version available at http://arxiv.org/abs/1604.03277
19. Doerr, B., Doerr, C., Yang, J.: k-bit mutation with self-adjusting k outperforms standard bit mutation. In: Proc. of Parallel Problem Solving from Nature (PPSN'16), *Lecture Notes in Computer Science*, vol. 9921, pp. 824–834. Springer (2016)
20. Doerr, B., Goldberg, L.A.: Drift analysis with tail bounds. In: Proc. of Parallel Problem Solving from Nature (PPSN'10), *Lecture Notes in Computer Science*, vol. 6238, pp. 174–183. Springer (2010)
21. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. Algorithmica **65**, 224–250 (2013)
22. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. Theoretical Computer Science **425**, 17–33 (2012)
23. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Mutation rate matters even when optimizing monotonic functions. Evolutionary Computation **21**, 1–27 (2013)
24. Doerr, B., Johannsen, D., Kötzing, T., Lehre, P.K., Wagner, M., Winzen, C.: Faster black-box algorithms through higher arity operators. In: Proc. of Foundations of Genetic Algorithms (FOGA'11), pp. 163–172. ACM (2011)
25. Doerr, B., Johannsen, D., Kötzing, T., Neumann, F., Theile, M.: More effective crossover operators for the all-pairs shortest path problem. Theoretical Computer Science **471**, 12–26 (2013)
26. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
27. Doerr, B., Künnemann, M.: Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm—different asymptotic runtimes for different instances. Theoretical Computer Science **561**, 3–23 (2015)
28. Doerr, B., Theile, M.: Improved analysis methods for crossover-based algorithms. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'09), pp. 247–254. ACM (2009)
29. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theoretical Computer Science **276**, 51–81 (2002)
30. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory of Computing Systems **39**, 525–544 (2006)
31. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Transactions on Evolutionary Computation **3**, 124–141 (1999)
32. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer Verlag (2003)

33. Erdős, P., Rényi, A.: On two problems of information theory. Magyar Tudományos Akadémia Matematikai Kutató Intézet Közleményei **8**, 229–243 (1963)
34. Fischer, S., Wegener, I.: The Ising model on the ring: Mutation versus recombination. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'04), *Lecture Notes in Computer Science*, vol. 3102, pp. 1113–1124. Springer (2004)
35. Gießen, C., Witt, C.: Population size vs. mutation strength for the (1+λ) EA on OneMax. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'15), pp. 1439–1446. ACM (2015)
36. Goldman, B.W., Punch, W.F.: Parameter-less population pyramid. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'14), pp. 785–792. ACM (2014)
37. Hansen, N., Gawelczyk, A., Ostermeier, A.: Sizing the population with respect to the local progress in (1,λ)-evolution strategies - a theoretical analysis. In: Proc. of the IEEE Congress on Evolutionary Computation (CEC'95), pp. 80–85. IEEE (1995)
38. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artificial Intelligence **127**, 57–85 (2001)
39. Jägersküpper, J.: Rigorous runtime analysis of the (1+1) ES: 1/5-rule and ellipsoidal fitness landscapes. In: Proc. of Foundations of Genetic Algorithms (FOGA'05), *Lecture Notes in Computer Science*, vol. 3469, pp. 260–281. Springer (2005)
40. Jansen, T.: Analyzing Evolutionary Algorithms—The Computer Science Perspective. Springer (2013)
41. Jansen, T., De Jong, K.A., Wegener, I.: On the choice of the offspring population size in evolutionary algorithms. Evolutionary Computation **13**, 413–440 (2005)
42. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms - a proof that crossover really can help. Algorithmica **34**, 47–66 (2002)
43. Jansen, T., Wegener, I.: On the analysis of a dynamic evolutionary algorithm. J. Discrete Algorithms **4**, 181–199 (2006)
44. Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter control in evolutionary algorithms: Trends and challenges. IEEE Transactions on Evolutionary Computation **19**, 167–187 (2015)
45. Kern, S., Müller, S.D., Hansen, N., Büche, D., Ocenasek, J., Koumoutsakos, P.: Learning probability distributions in continuous evolutionary algorithms - a comparative review. Natural Computing **3**, 77–112 (2004)
46. Kötzing, T.: Concentration of first hitting times under additive drift. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'14), pp. 1391–1398. ACM (2014)
47. Lässig, J., Sudholt, D.: Adaptive population models for offspring populations and parallel evolutionary algorithms. In: Proc. of Foundations of Genetic Algorithms (FOGA'11), pp. 181–192. ACM (2011)
48. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**, 623–642 (2012)
49. Mironovich, V., Buzdalov, M.: Hard test generation for maximum flow algorithms with the fast crossover-based evolutionary algorithm. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'15) (Companion Material), pp. 1229–1232. ACM (2015)
50. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing? In: Proceeding of the 7th Neural Information Processing Systems Conference (NIPS'93), *Advances in Neural Information Processing Systems*, vol. 6, pp. 51–58. Morgan Kaufmann (1993)
51. Oliveto, P.S., Lehre, P.K., Neumann, F.: Theoretical analysis of rank-based mutation - combining exploration and exploitation. In: Proc. of Congress on Evolutionary Computation (CEC'09), pp. 1455–1462. IEEE (2009)
52. Oliveto, P.S., Yao, X.: Runtime analysis of evolutionary algorithms for discrete optimization. In: A. Auger, B. Doerr (eds.) Theory of Randomized Search Heuristics, pp. 21–52. World Scientific Publishing (2011)
53. Raab, M., Steger, A.: "Balls into bins" – a simple and tight analysis. In: Proc. of Randomization and Approximation Techniques in Computer Science (RANDOM'98), *Lecture Notes in Computer Science*, vol. 1518, pp. 159–170. Springer (1998)
54. Rechenberg, I.: Evolutionsstrategie. Friedrich Fromman Verlag (Günther Holzboog KG), Stuttgart (1973)

55. Schumer, M.A., Steiglitz, K.: Adaptive step size random search. IEEE Transactions on Automatic Control **13**, 270–276 (1968)
56. Sudholt, D.: Crossover is provably essential for the Ising model on trees. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'05), pp. 1161–1167. ACM Press (2005)
57. Sudholt, D.: Crossover speeds up building-block assembly. In: Proc. of Genetic and Evolutionary Computation Conference (GECCO'12), pp. 689–702. ACM (2012)
58. Wegener, I.: Theoretical aspects of evolutionary algorithms. In: F. Orejas, P.G. Spirakis, J. van Leeuwen (eds.) Proc. of the 28th International Colloquium on Automata, Languages and Programming (ICALP'01), *Lecture Notes in Computer Science*, vol. 2076, pp. 64–78. Springer (2001)
59. Wegener, I.: Methods for the analysis of evolutionary algorithms on pseudo-Boolean functions. In: R. Sarker, M. Mohammadian, X. Yao (eds.) Evolutionary Optimization, pp. 349–369. Kluwer (2002)
60. Wegener, I.: Simulated annealing beats metropolis in combinatorial optimization. In: Proc. of International Colloquium on Automata, Languages and Programming (ICALP'05), *Lecture Notes in Computer Science*, vol. 3580, pp. 589–601. Springer (2005)
61. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Combinatorics, Probability & Computing **22**, 294–318 (2013)
62. Witt, C.: Fitness levels with tail bounds for the analysis of randomized search heuristics. Information Processing Letters **114**, 38–41 (2014)
63. Zarges, C.: Rigorous runtime analysis of inversely fitness proportional mutation rates. In: Proc. of Parallel Problem Solving from Nature (PPSN'08), *Lecture Notes in Computer Science*, vol. 5199, pp. 112–122. Springer (2008)
64. Zarges, C.: On the utility of the population size for inversely fitness proportional mutation rates. In: Proc. of Foundations of Genetic Algorithms (FOGA'09), pp. 39–46. ACM (2009)