Optimal parameter choices via precise black-box analysis

Benjamin Doerr, Carola Doerr, Jing Yang

Optimal Parameter Choices via Precise Black-Box Analysis*

Benjamin Doerr
École Polytechnique, CNRS
LIX - UMR 7161
91120 Palaiseau
France

Carola Doerr
Sorbonne Université, CNRS
Laboratoire d'informatique de Paris 6, LIP6
75252 Paris
France

Jing Yang
École Polytechnique, CNRS
LIX - UMR 7161
91120 Palaiseau
France

October 18, 2018

Abstract

It has been observed that some working principles of evolutionary algorithms, in particular, the influence of the parameters, cannot be understood from results on the asymptotic order of the runtime, but only from more precise results. In this work, we complement the emerging topic of precise runtime analysis with a first precise complexity theoretic result. Our vision is that the interplay between algorithm analysis and complexity theory becomes a fruitful tool also for analyses more precise than asymptotic orders of magnitude.

As particular result, we prove that the unary unbiased black-box complexity of the OneMax benchmark function class is $n \ln(n) - cn \pm o(n)$ for a constant c which is between 0.2539 and 0.2665. This runtime can be achieved with a simple (1+1)-type algorithm using a fitness-dependent mutation strength. When translated into the fixed-budget perspective, our algorithm finds solutions which are roughly 13% closer to the optimum than those of the best previously known algorithms. To prove our results, we formulate several new versions of the variable drift theorems, which also might be of independent interest.

1 Introduction

An important goal of the theory of randomized search heuristics (RSHs) is to prove mathematically founded statements about optimal parameter choices for these algorithms. The area of runtime analysis has contributed to this goal with rigorous analyses showing how the runtime of RSHs depends on one or more of their parameters. Unfortunately, due to the inherent difficulty of obtaining mathematically proven performance guarantees for RSHs, the majority of the existing runtime analyses only determine the asymptotic order of magnitude of the runtime (that is, the runtime in big-Oh notation). Naturally, such results usually give recommendations on optimal parameters again precise only up to the asymptotic order of magnitude, which for practical uses is often not precise enough. Only recently, made possible by the advancement of the analytical tools in the last 20 years, a number of results appeared that also make the

*A preliminary version of this work was presented at the *Genetic and Evolutionary Computation Conference (GECCO)* 2016 [DDY16b].

leading constant precise or even some lower order terms. These *precise runtime analyses* usually allowed much more precise statements about the ideal parameter choice.

In this work, we shall undertake the first steps to complement precise runtime analysis with equally precise complexity theoretic results. This is inspired by the area of classic algorithms theory, where the interplay between algorithm analysis and complexity theory has led to great advances. As in previous works on complexity theory for evolutionary algorithms, we build on the notion of *black-box complexity* introduced in the seminal work [DJW06]. In very simple words, the black-box complexity of an optimization problem is the number of fitness evaluations that suffice to find an optimal solution. This number is witnessed by a theoretically best-possible black-box algorithm, which may or may not be an evolutionary algorithm. In the former case, we have a proof that no better evolutionary algorithm exists, in the latter, one may ask the question why the existing evolutionary algorithms fall behind the theoretical optimum. As in classic algorithms, this can be a trigger to improve the existing algorithms. The example of [DDE15] shows that a careful analysis of the theoretically optimal black-box algorithm can also guide the design of new evolutionary algorithms.

1.1 Previous Works on Precise Runtime Analyses

As said above, the inherent difficulty of proving runtime guarantees for evolutionary algorithms and other RSHs by mathematical means for a long time prohibited runtime results that are more precise than giving the asymptotic order of magnitude. In fact, we note that for many simple optimization problems and very simple evolutionary algorithms, we even do not know the asymptotic order of the runtime. Examples include the runtime of the $(1+1)$ EA on the minimum spanning tree problem [Wit14] and on the single-source shortest path problem with the natural single-criterion fitness function [BBD+09], or the runtime of the global SEMO algorithm for the multi-objective LeadingOnesTrailingZeros problem [DKV13].

Before describing the few known precise runtime results, let us argue that in evolutionary computation precise results are more important than in classic algorithms theory. The performance measure in classic algorithms theory is the number of elementary operations performed in a run of the algorithm. Since the different elementary operations have different execution times and since further these are dependent on the hardware used, this performance measure can never give meaningful results that are more precise than the asymptotic order of magnitude. In evolutionary computation, the performance measure is the number of fitness evaluations. For this implementation-independent measure, results more precise than asymptotic orders are meaningful since we can often expect that each fitness evaluation takes more or less the same time and since we usually assume that the total runtime is dominated by the time spent on fitness evaluations.

Despite this interest in precise runtime estimates, only few results exist which determine a runtime precisely, that is, show upper and lower bounds that have at least the same leading constant. For classic evolutionary algorithms, the following is known.

A very early work on precise runtime analysis, apparently overlooked by many subsequent works, is the very detailed analysis on how the $(1+1)$ evolutionary algorithm ($(1+1)$ EA) with general mutation rate c/n, c a constant, optimizes the Needle and OneMax functions [GKS99]. Disregarding here the results on the runtime distributions, this work shows that the $(1+1)$ EA with mutation rate c/n finds the optimum of the Needle function in $(1+o(1))\frac{1}{1-e^{-c}}2^n$ time.

For OneMax, the runtime estimate in [GKS99] is $(1+o(1))\frac{e^c}{c}n\ln(n)$, more precisely, $\frac{e^c}{c}n\ln(n)\pm O(n)$. The proof of the latter result uses several deep tools from probability theory, among them Laplace transforms and renewal theory. Note that the main difficulty is the lower

bound. For a simple proof of the upper bound $(1+o(1))en\ln(n)$ for mutation rate $1/n$, see, e.g., [DJW02]. The first proofs of a lower bound of order $(1-o(1))en\ln(n)$ using more elementary methods were given, independently, in [DFW10,Sud13]. An improvement to $en\ln(n)-O(n)$ was presented in [DFW11]. Very recently, a very precise analysis of the expected runtime, specifying all lower order terms larger than $\Theta(\log(n)/n)$ with precise leading constants, was given in [HPR+18].

That the runtime bound of $(1+o(1))\frac{e^c}{c}n\ln(n)$ holds not only for OneMax, but any linear pseudo-Boolean function, was shown in [Wit13], ending a long quest for understanding this problem [DJW02,HY04,Jäg08,DJW12]. An extension of the OneMax result to $(1+\lambda)$ EAs was obtained in [GW15]. The bound of $(1+o(1))(\frac{e^c}{c}n\ln(n)+n\lambda\ln\ln(\lambda)/2\ln(\lambda))$ fitness evaluations contains the surprising result that the mutation rate is important for small offspring population sizes, but has only a lower-order influence once λ is sufficiently large (at least when restricted to mutation rates in the range $\Theta(1/n)$; note that [DGWY17, Lemma 1.2] indicates that mutation rates of a larger order of magnitude can give asymptotically smaller runtimes for larger values of λ). Results from [DNDD+18] imply that runtime of the $(1+1)$ EA with mutation rate $1/n$ remains $(1+o(1))en\ln(n)$ when the EA has to cope with dynamic changes of the optimum moving the optimum by an expected distance of $c\ln\ln(n)/n$, c a sufficiently small constant, independently in each iteration. To see this result, put $t=n$ in Theorem 5 of [DNDD+18] and note that the probability p_D for a dynamic change always is at most the expected change E_D. When calling this a precise runtime result, we tacitly assume that the known lower bound of $(1-o(1))en\ln(n)$ holds in the dynamic setting as well, as it is hard to believe that the $(1+1)$ EA should profit from random dynamic moves of the optimum.

In parallel independent work, the precise expected runtime of the $(1+1)$ EA on the LeadingOnes benchmark function was determined in [BDN10,Sud13] (note that [Sud13] is the journal version of a work that appeared at the same conference as [BDN10]). The work [Sud13] is more general in that it also regards the $(1+1)$ EA with Best-of-μ initialization (instead of just initializing with a random individual), the approach of [BDN10] has the advantage that it also allows to determines the distribution of the runtime (this was first noted in [DJWZ13] and was formally proven in [Doe18a]). The work [BDN10] also shows that the often recommended mutation rate of $p=1/n$ is not optimal. A runtime smaller by 16% can be obtained from taking $p=1.59/n$ and another 12% can be gained by using a fitness-dependent mutation rate. For several hyper-heuristics having the choice between the 1-bit flip and the 2-bit-flip operators, precise runtimes have been determined recently [LOW17,DLOW18]. These results in particular show that the classic selection hyper-heuristics *simple random*, *random gradient*, *greedy*, and *permutation* all have an inferior runtime to variants of the *random gradient* heuristic which use a chosen operator for a longer period τ than just until the first time no improvement is found. Since all heuristics regarded have a runtime of asymptotic order $\Theta(n^2)$, this runtime distinction would not have been possible without results making the leading constant precise.

In [DLMN17], the runtime of the $(1+1)$ EA on Jump functions was determined precise up to the leading constant. This showed that the optimal mutation rate for jump size $k=O(1)$ is $(1+o(1))k/n$, significantly larger than the common recommendation $1/n$. Also, this work led to the development of a heavy-tailed mutation operator which gave a uniformly good performance for all jump sizes k.

With very different methods, a precise runtime analysis for Plateau functions was conducted in [AD18]. The plateau function with radius k is similar to the jump function with parameter k, the difference being that now a plateau of equal fitness of radius k around the optimum is the difficult part of the search space. The precise runtime of the $(1+1)$ EA with arbitrary unbiased mutation operator (flipping one bit with at least some constant probability) is $(1+o(1))\binom{n}{k}p_{1:k}$, where $p_{1:k}$ denotes the probability that the mutation operator flips between

one and k bits. This implies again that a larger mutation rate than usual is optimal, namely $\sqrt[k]{k}/n \approx k/en$.

In summary, there are not too many results determining precise runtimes, but the ones we have significantly increased out understanding of how evolutionary algorithms work and what are good parameter values.

1.2 Black-Box Complexity

The notion of black-box complexity was introduced in [DJW06] with the goal of establishing a complexity theory for evolutionary algorithms. Complexity theory means that we aim at understanding the difficulty of a problem, that is, how well the best algorithm (from a specified class) can solve this problem. Since it is hard to provide a mathematically sound definition of what an evolutionary algorithm is, black-box complexity regards instead the (larger) class of black-box algorithms, that is, all algorithms which have access to the problem instance only via evaluating solution candidates. Consequently, the black-box complexity of a problem is the smallest number of fitness evaluation such that there is a black-box optimization algorithm solving all instances of the problem with at most this expected number of fitness evaluations.

It was observed early that the class of all black-box algorithms is significantly larger than the class of evolutionary algorithms. This led to some unexpectedly low black-box complexities witnessed by highly artificial algorithms. To develop a more suitable complexity theory, but also to study the effect of particular restrictions, several restricted notions of black-box complexity have been suggested, among them unbiased black-box complexity (admitting only algorithms which generate solution candidates from previous solutions in an unbiased fashion) [LW12], memory restricted black-box complexity (allowing to store only a certain number of solution candidates) [DJW06, DW12], and ranking-based black-box complexity (allowing only to compare fitness values, but not to exploit absolute fitness values) [FT11, DW14]. We refer to the survey [Doe18b] for a detailed discussion of these and further black-box complexity notions.

In this work, we build on the unary unbiased black-box complexity [LW12], which is the most appropriate model for mutation-based search heuristics. In simple words, a unary unbiased black-box algorithm for the optimization of a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is allowed (i) to sample search points uniformly at random from the search space $\{0,1\}^n$, and (ii) to generate new search points from applying unbiased mutation operators to previously found search points. Here *unbiased* means that the operator is invariant under automorphisms of the hypercube $\{0,1\}^n$. In other words, the operator has to treat both the bit-positions and the bit-values in a symmetric fashion. For unary operators (usually called *mutation operators*), we give a simple characterization of the set of unbiased operators in Lemma 1, roughly saying that an unbiased mutation operator always can be described via first sampling a number $r \in [0..n] := \{0, \dots, n\}$ according to a given probability distribution and then flipping a set of r bits chosen uniformly at random from all r-sets of bits. Furthermore, in an unbiased algorithm all selection operations may only rely on the observed fitness values, but not on the particular representations of the solutions.

While we claim that this work is the first significant progress towards a useful precise complexity theory for evolutionary algorithms, we note that our work is not the first to show precise black-box complexity results. In [DJW06], the unrestricted black-box complexity of the Needle and Trap function classes were shown to be $(2^n+1)/2$ (Theorem 1 and Proposition 3), that of the BinaryValue class was shown to be $2 - 2^{-n}$ (Theorem 4), and that of the not-permuted LeadingOnes class was proven to be $(1 \pm o(1))n/2$ (Theorem 6). In [DKLW13, Theorem 16], it was shown that the unrestricted black-box complexity of the multi-criteria formulation of the single-source shortest path problem is $n-1$, where n denotes the number of vertices of the

input graph. In [BDK16], the unrestricted black-box complexity of extreme JUMP functions, that is, with jump size that large that only the middle (for n even) or the two middle (for n odd) Hamming levels are visible was determined to be $n + \Theta(\sqrt{n})$. Since all these result regard unrestricted black-box complexities and some find artificially small complexities, we do not feel that these results contribute immediately towards a precise black-box complexity theory that can help to improve the evolutionary algorithms currently in use.

1.3 Overview of Our Results

With the goal of starting a complexity theory targeting precise results, we analyze the unary unbiased black-box complexity of the ONEMAX benchmark function

$$\text{OM} : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^{n} x_i.^{1}$$

It is known that the unary unbiased black-box complexity of ONEMAX is of order $\Theta(n \log n)$ [LW12]. The simple *randomized local search* (RLS) heuristic, a hill-climber flipping single random bits, is easily seen to have a runtime of at most $n \ln(n) + \gamma n + \frac{1}{2} \approx n \ln(n) + 0.5772n$ where $\gamma = 0.5772\ldots$ is the Euler-Mascheroni constant. This can be shown by a reduction to the coupon collector problem. The precise complexity [DD16] of this algorithm is

$$n \ln(n) + (\gamma - \ln(2))n + o(1) \approx n \ln n - 0.1159n.$$

Using a best-of-μ initialization rule with a suitably chosen μ, the running time of RLS decreases by an additive $\Theta(\sqrt{n \log n})$ term [dPdLDD15]. Prior to the present work, this was the best unary unbiased algorithm known for ONEMAX, and thus its time complexity was the best known upper bound for the unary unbiased black-box complexity of this function.

In this work, we show that the unary unbiased black-box complexity of ONEMAX is

$$n \ln(n) - cn \pm o(n)$$

for a constant c for which we show $0.2539 < c < 0.2665$. We also show how to numerically compute this constant with arbitrary precision. More importantly, our analysis reveals (and needs) a number of interesting structural results which enlarge our general understanding and which might find applications in other algorithm analyses in the future. In particular, we observe the following, which we will discuss in more detail in the following subsections.

1. Drift-maximization is near-optimal: On ONEMAX, any unary unbiased algorithm which in each iteration (by selecting a suitable parent and choosing a suitable mutation operator) maximizes the expected fitness gain over the best-so-far individual ("drift") has essentially an optimal runtime. More precisely, its runtime exceeds the optimal one (the unary unbiased black-box complexity) by at most $O(n^{2/3} \log^9 n)$.

2. There is such a drift-maximizing algorithm which selects in each iteration the best-so-far solution and mutates it by flipping a fixed number of bits. This number depends solely on the fitness of the current-best solution. It is always an odd number and it decreases with increasing fitness.

[1] The reader not familiar with black-box complexity may wonder that this is a single function, whose optimum is of course known to be the string $(1, \ldots, 1)$. However, we are interested in the time needed by a unary unbiased algorithm—which cannot simply query this string but needs to construct it from unary unbiased operators—to find this string. Note further that the performance of any such algorithm is identical on all functions having a fitness landscape isomorphic to that of OM. That is, for every $z \in \{0,1\}^n$ and every unary unbiased algorithm A, the performance of A on $f_z : \{0,1\}^n \to \{0,1,...,n\}, x \mapsto |\{i \in [n] | x_i = z_i\}|$ is identical to its performance on $\text{OM} = f_{(1,...,1)}$.

3. In the language of fixed-budget computation introduced by Jansen and Zarges [JZ14], the drift-maximizing algorithm (and thus also any optimal unary unbiased black-box algorithm) computes solutions with expected fitness distance to the optimum roughly 13% smaller than the previous-best algorithms (RLS, RLS with best-of-μ initialization).

To show these results, we use a number of technical tools which might be suitable for other analyses as well. Among them are simplified (but sightly stronger) variants of the variable drift theorems for discrete search spaces and versions of the lower bound variable drift theorem which can tolerate large progresses if these happen with sufficiently small probability.

1.4 Maximizing Drift is Near-Optimal

Evolutionary algorithms build on the idea that iteratively maximizing the fitness is a good approach. This suggests to try to generate the offspring in a way that the expected fitness gain over the best-so-far solution is maximized. Clearly, this is not a successful idea for each and every problem, as easily demonstrated by examples like the DISTANCE and the TRAP functions [DJW02], where the fitness leads the algorithm into a local optimum, or the difficult-to-optimize monotonic functions constructed in [DJS+13, LS18, Len18], where the fitness leads to the optimum, but via a prohibitively long trajectory. Still, one might hope that for problems with a good fitness-distance correlation (and OM has the perfect fitness-distance correlation), maximizing the expected fitness gain is a good approach. This is roughly what we are able to show.

More precisely, we cannot show that maximizing the expected fitness gain leads to the optimal unary unbiased black-box algorithm. It turns out this is also not true, even if we restrict ourselves to elitist algorithms, which cannot use tricks like minimizing the fitness and inverting the search point once the all-zero string was found. In fact, the elitist algorithm flipping in each iteration the number of bits that minimizes the expected remaining runtime is different from the drift-maximizing one. For all realistic problem sizes, however, the differences between the expected runtime of our drift maximizer and that of the optimal elitist algorithm are negligibly small [BD18].

What we can prove, however, is that the algorithm which in each iteration takes the best-so-far solution and applies to it the unary unbiased mutation operator maximizing the expected fitness gain has an expected optimization time which exceeds the unary unbiased black-box complexity by at most an additive term of order $O(n^{2/3} \log^9 n)$.

We note that this result, while natural, is quite difficult to obtain and relies on a number of properties particular to this process, in particular, the fact that we have a good structural understanding of the maximal drift.

1.5 Maximizing the Drift via the Right Fitness-Dependent Mutation Strength

Once we decided how to choose the parent individual, in principle it is easy to mutate it in such a way that the drift is maximized. Since any unary unbiased mutation operator can be seen as a convex combination of r-bit flip operators, $r \in [0..n]$, and since the drift stemming from such an operator is the corresponding convex combination of the drifts stemming from these operators, we only need to determine, depending on the fitness of the parent, a value for r such that flipping r random bits maximizes the fitness gain over the parent. For a concrete value of n and a concrete fitness of the parent, one can compute the best value of r in time $O(n^2)$.

We need some more mathematical arguments to (i) obtain a structural understanding of the optimal r-value and (ii) to obtain a runtime estimate valid for all values of n. To this

aim, we shall first argue that when the fitness distance d of the parent from the optimum is at most $(\frac{1}{2} - \varepsilon)n$ for an arbitrarily small constant ε, then the optimal drift is obtained from flipping a constant number r of bits (Lemma 20). For any constant r, the drift obtained from flipping r bits can be well approximated by a degree r polynomial in the relative fitness distance d/n (Theorem 22). By this, we overcome the dependence on n, that is, apart from this small approximation error we can, by regarding these polynomials, determine a function $\tilde{R}_{\text{opt}} : [0, \frac{1}{2} - \varepsilon] \to \mathbb{N}$ such that for all $n \in \mathbb{N}$ and all $d \in [0..(\frac{1}{2} - \varepsilon)n]$ the near-optimal number of bits to flip (that is, optimal apart from the approximation error) is $\tilde{R}_{\text{opt}}(d/n)$. This $\tilde{R}_{\text{opt}}$ is decreasing, that is, the closer we are to the optimum, the smaller is the near-optimal number of bits to flip. Interestingly, $\tilde{R}_{\text{opt}}$ is never even, so the near-optimal number of bits to flip is always odd (the same holds for the truly optimal number of bits, as we also show). These (and some more) structural properties allow to compute numerically the interval in which flipping r bits is near-optimal (for all odd r). From these we estimate the runtime by numerically approximating the integral describing the runtime via the resulting drifts. This gives approximations for both the runtime of our drift-maximizer and the unary unbiased black-box complexity, which are precise apart from an arbitrary small $O(n)$ term.

We note that previous works have studied drift maximizing variants of RLS and the (1+1) EA by empirical means. For $n = 100$, Bäck [Bäc92] computed the drift-maximizing mutation rates for different $(1 + \lambda)$ and $(1, \lambda)$ EAs. Fialho and co-authors considered in [FCSS08, FCSS09] for $n = 1,000$ and $n = 10,000$, respectively, $(1 + \lambda)$-type RLS-variants which choose between flipping exactly 1, 3, or 5 bits or applying standard bit mutation with mutation rate $p = 1/n$. A simple empirical Monte Carlo evaluation is conducted to estimate the average progress obtained by any of these four operators. None of the three mentioned works, however, further investigates the difference between the drift maximizing algorithm and the optimal (i.e., time-minimizing) one.

1.6 Fixed-Budget Result

Computing the runtime of our drift-maximizing algorithm, we observe that the fitness-dependent mutation strength gives a smallish-looking improvement of roughly $0.14n$ in the $\Theta(n \log n)$ runtime. However, if we view our result in the fixed-budget perspective [JZ14], then (after using the Azuma inequality in the martingale version to show sufficient concentration) we see that if we take the expected solution quality after a fixed number (budget) of iterations as performance measure, then our algorithm gives a roughly 13% smaller fitness distance to the optimum compared to the previous-best algorithm (provided that the budget is at least $0.2675n$).

1.7 Methods

To obtain our results, we use a number of methods which might find applications is other precise runtime and black-box complexity analyses. Among these, we want to highlight here a few new versions of the variable drift theorems.

A simple, but useful variant of Johannsen's upper bound drift theorem can be obtained for processes on the non-negative integers. Here the expected runtime can be simply written as the sum of the reciprocals of the lower bounds on the drift (Theorem 7). This avoids the use of integrals as in Johannsen's formulation [Joh10]. In our application, we profit from the fact that the new theorem is more precise as it avoids the error stemming from approximating the discrete drift via a continuous progress estimate (this error is usually small, but for our purposes the resulting error of order $\Theta(n)$ would be too large).

For the lower bound variable drift theorem from [DFW11], besides a similar simplification for processes on the non-negative integers, we add two improvements. The first concerns the

requirement that the process may not make too large progress in a single step. While it is clear that some such condition is necessary for the drift result to hold, the strict condition of [DFW11] is difficult to enforce in evolutionary computation. When, e.g., using standard-bit mutation, large progresses are highly unlikely (simply because the number of bits that flip is small), but large progresses cannot be ruled out completely. For this reason, we devise a variant that can deal with such situations. Our relaxed condition is that large progresses occur only with small probability. We note that a similar drift theorem was given in [GW16, Theorem 2], however, it appears to be more technical than our version.

A second difficulty with the drift theorem in [DFW11] (and likewise with the one in [GW16]) is the fact that a function h has to be given such that the drift from the point x can be bounded from above by $h(c(x))$, where c is the upper bound for the progress possible from x. This is significantly less convenient than just finding a bound $h(x)$ for the drift from point x as in the upper bound drift theorem. Therefore we formulate in Theorem 10 a variant of the lower bound drift theorem which also only requires an upper bound $h(x)$ for the drift from x. While not a deep result, we expect that this version eases the process of finding lower bounds via variable drift in the future.

1.8 Connection to Parameter Control

As discussed in Section 1.5 the drift-maximizing algorithm uses mutation strengths that depend on the fitness value of a current-best solution. These drift-maximizing mutation strengths change from flipping half the bits (for search points with fitness $n/2$) to flipping single bits (for search points close to the optimal solution). The algorithm is therefore an example for a state-dependent parameter controlled EA, in the taxonomy presented in [DD18]. For reasons of space and focus, we do not give an extended introduction to dynamic parameter choices here in this work, but refer the interested reader to the recent book chapter [DD18] instead. We note, however, that in [DDY16a], a work subsequent to the conference version of the present work [DDY16b], we have presented a learning-based control mechanism which tracks the drift-maximizing mutation strength so closely that its overall expected optimization time is at most an additive $o(n)$ term worse than that of the drift-maximizer.

2 Problem Setting and Useful Tools

In this section we briefly describe the black-box setting regarded in this work, the unary unbiased model proposed by Lehre and Witt [LW12]. The variation operators that are admissible in this model are characterized in Lemma 1. We also collect (Section 2.2) a number of drift theorems that we build upon in our mathematical analysis. In Theorems 7 to 10 we present new versions of the variable drift theorems which we will use.

2.1 The Unary Unbiased Black-Box Setting

The main goal of our work is to determine a precise bound for the unary unbiased black-box complexity of OneMax, the problem of maximizing the function $\textsc{Om} : \{0,1\}^n \to \mathbb{R}, x \mapsto \sum_{i=1}^n x_i$, which assigns to each bit string the number of ones in it. That is, we aim at identifying a best-possible mutation-based algorithm for this problem. The unary unbiased black-box complexity of OneMax is the smallest expected number of function evaluations that any algorithm following the structure of Algorithm 1 exhibits on this problem before and including the first

Algorithm 1: Blueprint of a unary unbiased black-box algorithm

1 **Initialization:**
2 $t \leftarrow 0$;
3 Choose $x(t)$ uniformly at random from $S = \{0,1\}^n$;
4 **Optimization:**
5 **repeat**
6 $t \leftarrow t+1$;
7 Evaluate $f(x(t-1))$;
8 Based on $\Big(f(x(0)), \dots, f(x(t-1))\Big)$ choose a probability distribution p_s on $[0..t-1]$ and a unary unbiased operator $(p(\cdot|x))_{x\in\{0,1\}^n}$;
9 Randomly choose an index i according to p_s;
10 Generate $x(t)$ according to $p(\cdot|x(i))$;
11 **until** termination condition met;

evaluation of the unique global optimum $(1,\dots,1)$.[2] In line 9 of Algorithm 1 a unary unbiased variation operator is asked for. In the context of optimizing pseudo-Boolean functions, a *unary* operator is an algorithm that is build upon a family $(p(\cdot \mid x))_{x\in\{0,1\}^n}$ of probability distributions over $\{0,1\}^n$. For a given input x, the operator outputs a new string that it samples from the distribution $p(\cdot \mid x)$. A unary operator is *unbiased* if all members of its underlying family of probability distributions are symmetric with respect to the bit positions $[n] := \{1,\dots,n\}$ and the bit values 0 and 1 (cf. [LW12, Doe18b] for a discussion). Unary unbiased variation operators are also referred to as *mutation operators*.

The characterization of unary unbiased variation operators in Lemma 1 below states that each such operator is uniquely defined via a probability distribution r_p over the set $[0..n] := \{0\} \cup [n]$ describing how many bits (chosen uniformly at random without replacement) are flipped in the argument x to create a new search point y. Finding a best possible mutation-based algorithm is thus identical to identifying an optimal strategy to select the distribution r_p. This characterization can also be derived from [DKLW13, Proposition 19], although the original formulation of Proposition 19 in [DKLW13] requires as search space $[n]^{n-1}$.

Lemma 1. *For every unary unbiased variation operator $(p(\cdot|x))_{x\in\{0,1\}^n}$ there exists a probability distribution r_p on $[0..n]$ such that for all $x, y \in \{0,1\}^n$ the probability $p(y|x)$ that $(p(\cdot|x))_{x\in\{0,1\}^n}$ samples y from x equals the probability of sampling a random number r from r_p and then flipping r bits in x to create y. On the other hand, each distribution r_p on $[0..n]$ induces a unary unbiased variation operator.*

To prove Lemma 1, we introduce the following notation.

Definition 2. *Let $r \in [0..n]$. For every $x \in \{0,1\}^n$ the operator flip_r creates an offspring y from x by selecting r positions $i_1,\dots,i_r$ in $[n]$ uniformly at random (without replacement), setting $y_i := 1-x_i$ for $i \in \{i_1,\dots,i_r\}$, and copying $y_i := x_i$ for all other bit positions $i \in [n]\setminus\{i_1,\dots,i_r\}$.*

With this notation, Lemma 1 states that every unary unbiased variation operator $(p(\cdot|x))_{x\in\{0,1\}^n}$ can be described by Algorithm 2, for a suitably chosen probability distribution r_p.

[2] In the interest of a concise discussion, we do not provide here an extended discussion of black-box complexity. Interested readers are referred to [Doe18b] for a summary of different black-box complexity models and known results.

Algorithm 2: The unary unbiased operator r_p samples for a given x an offspring by sampling from r_p the number r of bits to flip and then applying flip_r to x.

1 Choose an integer $r \in [0..n]$ according to r_p;
2 Sample $y \leftarrow \text{flip}_r(x)$;

Since it will be needed several times in the remainder of this work, we briefly recall the following simple fact about the expected OM-value of an offspring generated by flip_r.

Remark 3. *Let $x \in \{0,1\}^n$ and $r \in [0..n]$. The number of 1-bits that are flipped by the variation operator flip_r follows a hypergeometric distribution with expectation equal to $r\text{OM}(x)/n$. The expected number of 0-bits that are flipped by the operator flip_r equals $r(n - \text{OM}(x))/n$. The expected OM-value of an offspring generated from x by applying flip_r is thus equal to $\text{OM}(x) - r\text{OM}(x)/n + r(n - \text{OM}(x))/n = (1 - 2r/n)\text{OM}(x) + r$.*

To prove Lemma 1 we show the following.

Lemma 4. *For every unary unbiased variation operator $(p(\cdot|x))_{x\in\{0,1\}^n}$ there exists a exists a probability distribution r_p on $[0..n]$ such that for all $x, y \in \{0,1\}^n$*

$$p(y|x) = r_p(H(x,y))/\binom{n}{H(x,y)}, \tag{1}$$

where here and henceforth $H(x,y)$ denotes the Hamming distance of x and y.

Proof. Let p be a unary unbiased variation operator. We first show that for all $x, y_1, y_2 \in \{0,1\}^n$ with $H(x,y_1) = H(x,y_2)$, the equality $p(y_1|x) = p(y_2|x)$ holds. This shows that for any fixed Hamming distance $d \in [n]$ and every string x, the probability distribution on the d-neighborhood $\mathcal{N}_d(x) := \{y \in \{0,1\}^n, H(x,y) = d\}$ of x is uniform.

Using the fact that the bit-wise XOR operator $\oplus$ preserves the Hamming distance, we obtain that for any $y_1, y_2 \in \mathcal{N}_d(x)$ it holds that

$$H(x \oplus x, y_1 \oplus x) = H(x \oplus x, y_2 \oplus x) = d.$$

Since $x \oplus x = (0, \ldots, 0)$, we thus observe that

$$\sum_{i=1}^{n} (y_1 \oplus x)_i = \sum_{i=1}^{n} (y_2 \oplus x)_i = d.$$

This implies that there exists a permutation $\sigma \in S_n$ such that

$$\sigma(y_1 \oplus x) = y_2 \oplus x.$$

According to the definition of unary unbiased variation operators, p is invariant under "$\oplus$" and "σ", yielding

$$\begin{aligned} p(y_1|x) &= p(y_1 \oplus x | x \oplus x) \\ &= p(\sigma(y_1 \oplus x) | \sigma(x \oplus x)) \\ &= p(y_2 \oplus x | x \oplus x) = p(y_2|x). \end{aligned}$$

This shows that $p(\cdot|x)$ is uniformly distributed on $\mathcal{N}_d(x)$ for any $d \in [n]$ and any $x \in \{0,1\}^n$.

For every $x \in \{0,1\}^n$ and every $d \in [n]$ let $p_{(d,x)}$ denote the probability of sampling a specific point at distance d from x. That is, for $y_1 \in \mathcal{N}_d(x)$ we have $p(y_1|x) = p_{(d,x)}$. For $x' \neq x$ let y' denote $y_1 \oplus (x \oplus x')$. Then by the unbiasedness of p we obtain that

$$H(y', x') = H(y_1, x) = d, \quad \text{and } p_{(d,x')} = p(y'|x') = p(y' \oplus x \oplus x' \mid x' \oplus x \oplus x') = p(y_1|x) = p_{(d,x)}.$$

Thus, for all $x, x' \in \{0,1\}^n$ it holds that $p_{(d,x)} = p_{(d,x')} =: p_{(d)}$.

For the unary unbiased variation operator p we can therefore define a distribution r_p on $[0..n]$ by setting

$$r_p(d) = \binom{n}{d} p_{(d)}.$$

For all $x, y \in \{0,1\}^n$ we obtain

$$p(y|x) = p_{(d,x)} = p_{(d)} = r_p(d) / \binom{n}{d},$$

where we abbreviate $d := H(x,y)$. This shows the desired equation (1). □

2.2 Drift Analysis

The main tool in our work is drift analysis, a well-established method in the theory of randomized search heuristics. Drift analysis tries to translate information about the expected progress of an algorithm into information about the expected runtime, that is, the expected hitting time of an optimal solution. We note that typically drift theorems are phrased in a way that they estimate the time a stochastic process on a subset of the real numbers takes to hit zero. This is convenient when regarding as process some distance of the current-best solution of an evolutionary algorithm to the optimum. Of course, via elementary transformations all drift results can be rephrased to hitting times for other targets.

Drift analysis was introduced to the field in the seminal work of He and Yao [HY04]. They proved the following additive drift theorem, which assumes a uniform bound on the expected progress.

Theorem 5 (Additive drift theorem [HY04])**.** *Let $(X_t)_{t \geq 0}$ be a sequence of non-negative random variables over a finite state space in $\mathbb{R}$. Let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. If there exist $c, d > 0$ such that*

$$c \leq E\left(X_t - X_{t+1} \mid T > t\right) \leq d,$$

then

$$\frac{X_0}{d} \leq E\left(T \mid X_0\right) \leq \frac{X_0}{c}.$$

Since the progress of many algorithms slows down the closer they get to the optimum, the requirement of Theorem 5 to establish an *additive* bound on the drift is often not very convenient. For this reason, a *multiplicative* drift theorem that only requires that the expected progress is proportional to the distance was proposed in [DJW12]. The multiplicative drift theorem is often a good tool for the analysis of randomized search heuristic on the OneMax test function and related problems. However, for the very precise bound that we aim at it in this present work, it does not suffice to approximate the true drift behavior by a linear progress estimate. For this reason, we resort to the most general technique known in the area of drift analysis, which is called *variable* drift, and which assumes no particular behavior of the progress. Variable drift was independently developed in [MRC09] and [Joh10].

Theorem 6 (Johannsen's Theorem [Joh10]). *Let $(X_t)_{t\geq 0}$ be a sequence of non-negative random variables over a finite state space in $\mathcal{S} \subset \mathbb{R}$ and let $x_{\min} = \min\{x \in \mathcal{S} : x > 0\}$. Furthermore, let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose that there exists a continuous and monotonically increasing function $h : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that for all $t < T$ it holds that*

$$E\left(X_t - X_{t+1} \mid X_t\right) \geq h(X_t).$$

Then

$$E\left(T \mid X_0\right) \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X_0} \frac{1}{h(x)} dx.$$

We recall that a function $h : \mathbb{R} \to \mathbb{R}$ is monotonically increasing (decreasing), if $h(r) \leq h(s)$ ($h(r) \geq h(s)$, respectively) holds for all $r, s \in \mathbb{R}$ with $r < s$.

In Theorem 6 the assumption that h is continuous is not necessary. This was shown in [RS14] by redoing the original proof, but replacing the clumsy mean-value argument used to show equation (4.2.1) in [Joh10] by a natural elementary estimate. This result was further used to reprove a not so easily accessible variable drift theorem for discrete search spaces [MRC09]. We give below an alternative proof for this discrete drift theorem, which uses the elementary idea to approximate a step function by continuous functions.

Theorem 7 (Discrete Variable Drift, upper bound). *Let $(X_t)_{t\geq 0}$ be a sequence of random variables in $[0..n]$ and let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose that there exists a monotonically increasing function $h : \{1, \dots, n\} \to \mathbb{R}_0^+$ such that*

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

holds for all $t < T$. Then

$$E\left(T \mid X_0\right) \leq \sum_{i=1}^{X_0} \frac{1}{h(i)}.$$

Proof. For all $0 < \varepsilon < 1$, define a function $h_\varepsilon : [1, n] \to \mathbb{R}^+$ by

$$h_\varepsilon(x) := \begin{cases} h(\lfloor x \rfloor) + \frac{x - \lfloor x \rfloor}{\varepsilon}\big(h(\lceil x \rceil) - h(\lfloor x \rfloor)\big) & \text{for } 0 < x - \lfloor x \rfloor \leq \varepsilon, \\ h(\lceil x \rceil) & \text{for } \varepsilon < x - \lfloor x \rfloor, \end{cases} \tag{2}$$

The continuous monotone function h_ε satisfies $E(X_t - X_{t+1} \mid X_t) \geq h_\varepsilon(X_t)$ for all $t < T$. We compute

$$\begin{aligned} \int_1^{X_0} \frac{1}{h_\varepsilon(x)} dx &\leq \sum_{i=2}^{X_0} \frac{\varepsilon}{h(i-1)} + \frac{1-\varepsilon}{h(i)} \\ &= \frac{\varepsilon}{h(1)} - \frac{\varepsilon}{h(X_0)} + \sum_{i=2}^{X_0} \frac{1}{h(i)}. \end{aligned}$$

Hence by Theorem 6, we have $E(T \mid X_0) \leq \frac{\varepsilon}{h(1)} - \frac{\varepsilon}{h(X_0)} + \sum_{i=1}^{X_0} \frac{1}{h(i)}$ for all $\varepsilon > 0$, which proves the claim. □

To prove lower bounds on runtimes, the following variable drift theorem having a similar structure as Theorem 6 was given in [DFW11]. We state the result in the original formulation of [DFW11], but note that as for Theorem 6, the assumptions that h and c are continuous are not necessary. As a main difference to Theorem 6, we now have the additional requirement that the process with probability one in each round does not move too far towards the target (first condition of the theorem below).

Theorem 8 (Variable Drift, lower bound [DFW11]). *Let $(X_t)_{t\geq 0}$ be a decreasing sequence of non-negative random variables over a finite state space in $\mathcal{S} \subset \mathbb{R}$ and let $x_{\min} = \min\{x \in \mathcal{S} : x > 0\}$. Furthermore, let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose that there exists two continuous and monotonically increasing function $c, h : \mathbb{R}_0^+ \to \mathbb{R}_0^+$ such that*

1. *$X_{t+1} \geq c(X_t)$,*
2. *$E(X_t - X_{t+1} \mid X_t) \leq h(c(X_t))$.*

Then

$$E(T \mid X_0) \geq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{X_0} \frac{1}{h(x)} dx.$$

The first condition of the theorem above, that with probability one no progress beyond a given limit is made, is a substantial restriction to the applicability of the theorem, in particular, when working with evolutionary algorithms, where often any parent can give birth to any offspring (though usually with very small probability). In [DFW11], this problem was overcome with the simple argument that a progress of more than $\sqrt{x}$ from a search point with fitness distance x occurs with such a small probability that it does with high probability not occur in a run of typical length.

For the precise bounds that we aim at, such a simple argument cannot work. We therefore show a version of the lower bound variable drift theorem which allows larger jumps provided that they occur sufficiently rarely. To prove this result, we cannot use a blunt union bound over all bad events of too large jumps, but have to take these large jumps into account when computing the additive drift in a proof analogous to the one of [DFW11]. This idea was already used in [GW16], where a similar, but more technical drift theorem was derived. The result of [GW16] is valid for arbitrary domains, whereas we restrict ourselves to processes over the non-negative integers, but this is not the main reason for the simplicity of our result.

Theorem 9 (Discrete Variable Drift, lower bound). *Let $(X_t)_{t\geq 0}$ be a sequence of decreasing random variables in $[0..n]$ and let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose that there exists two monotonically increasing functions $c : [n] \to [0..n]$ and $h : [0..n] \to \mathbb{R}_0^+$, and a constant $0 \leq p < 1$ such that*

1. *$X_{t+1} \geq c(X_t)$ with probability at least $1 - p$ for all $t < T$,*
2. *$E(X_t - X_{t+1} \mid X_t) \leq h(c(X_t))$ holds for all $t < T$.*

Let $g : [0..n] \to \mathbb{R}_0^+$ be the function defined by $g(x) = \sum_{i=0}^{x-1} \frac{1}{h(i)}$. Then

$$E(T \mid X_0) \geq g(X_0) - \frac{g^2(X_0)p}{1 + g(X_0)p}.$$

Proof. The function g is strictly monotonically increasing. We have $g(X_t) = 0$ if and only if $X_t = 0$. Using condition 1 and the monotonicity of h, in the case $X_{t+1} \geq c(X_t)$ we have

$$g(X_t) - g(X_{t+1}) = \sum_{i=X_{t+1}}^{X_t-1} \frac{1}{h(i)} \leq \frac{X_t - X_{t+1}}{h(X_{t+1})} \leq \frac{X_t - X_{t+1}}{h(c(X_t))},$$

and otherwise

$$g(X_t) - g(X_{t+1}) \leq g(X_t) \leq g(X_0).$$

Using inequality $E(X_t - X_{t+1} \mid X_t) \leq E(X_t - X_{t+1} \mid X_{t+1} \geq c(X_t))$ and condition 2, we have

$$\begin{aligned} E\left(g(X_t) - g(X_{t+1}) \mid g(X_t)\right) &\leq E\left(\frac{X_t - X_{t+1}}{h(c(X_t))} \mid X_{t+1} \geq c(X_t)\right)(1-p) + g(X_0)p \\ &\leq 1 + g(X_0)p. \end{aligned}$$

Applying the additive drift theorem 5 to $g(X_t)_{t\geq 0}$, we obtain

$$E\left(T \mid X_0\right) = E\left(T \mid g(X_0)\right) \geq \frac{g(X_0)}{1+g(X_0)p} = g(X_0) - \frac{g^2(X_0)p}{1+g(X_0)p}.$$

□

To apply the drift theorem above (or Theorem 8) one needs to guess a suitable function h such that $h \circ c$ is an upper bound for the drift. The following simple reformulation overcomes this difficulty by making $h(x)$ simply an upper bound for the drift from a state x. This also makes the result easier to interpret. The influence of large jumps, as quantified by c, now is that the runtime bound is not anymore the sum of all $h(x)^{-1}$ as in the upper bound theorem, but of all $h(\mu(x))^{-1}$, where $\mu(x)$ is the largest point y such that $c(y) \leq x$. So in simple words, we have to replace the drift at x pessimistically by the largest drift among the points from which we can go to x (or further) in one round with probability more than p.

Theorem 10 (Discrete Variable Drift, lower bound)**.** *Let $(X_t)_{t\geq 0}$ be a sequence of decreasing random variables in $[0..n]$ and let T be the random variable that denotes the earliest point in time $t \geq 0$ such that $X_t = 0$. Suppose that there exists two functions $c : \{1, \dots, n\} \to [0..n]$ and monotonically increasing $h : [0..n] \to \mathbb{R}_0^+$, and a constant $0 \leq p < 1$ such that*

1. *$X_{t+1} \geq c(X_t)$ with probability at least $1-p$ for all $t < T$,*
2. *$E(X_t - X_{t+1} \mid X_t) \leq h(X_t)$ holds for all $t < T$.*

Let $\mu : [0..n] \to [0..n]$ be the function defined by $\mu(x) = \max\{i \mid c(i) \leq x\}$ and $g : [0..n] \to \mathbb{R}_0^+$ be the function defined by $g(x) = \sum_{i=0}^{x-1} \frac{1}{h(\mu(i))}$. Then

$$E(T \mid X_0) \geq g(X_0) - \frac{g^2(X_0)p}{1+g(X_0)p}.$$

Proof. Let $\hat{c} : [0..n] \to [0..n], r \mapsto \hat{c}(r) := \min\{i \mid \mu(i) \geq r\}$. By definition, $\hat{c}$ is monotonically increasing. Let $r \in [n]$. Since $r \in \{i \mid c(i) \leq c(r)\}$, we have $\mu(c(r)) \geq r$, which implies $c(r) \in \{i \mid \mu(i) \geq r\}$. Hence $\hat{c}(r) \leq c(r)$. This shows that we have $X_{t+1} \geq \hat{c}(X_t)$ for all $t < T$.

The definition of $\hat{c}$ implies that $\mu(\hat{c}(r)) \geq r$. Together with the monotonicity of h, we obtain $E(X_t - X_{t+1} \mid X_t) \leq h(X_t) \leq h(\mu(\hat{c}(X_t)))$. Let $\hat{h} : [0..n] \to \mathbb{R}_0^+, r \mapsto \hat{h}(r) := h(\mu(r))$. It is easy to see from the definition that μ is monotonically increasing, therefore, $\hat{h}$ is also monotonically increasing and it satisfies $E(X_t - X_{t+1} \mid X_t) \leq \hat{h}(\hat{c}(X_t))$. Applying Theorem 9 to $\hat{h}$ and $\hat{c}$ shows the claim. □

3 Maximizing Drift is Near-Optimal

The goal of this section is to show that the algorithm which maximizes the expected progress over the best-so-far search point is optimal, apart from lower-order terms $o(n)$, among all unary unbiased black-box algorithms. Consequently, its expected optimization time is essentially the unary unbiased black-box complexity.

Algorithm 3: Structure of our algorithm

```
Choose x uniformly at random from S = {0,1}^n;
for t = 1, 2, ... do
    y ← flip_{R(OM(x))}(x);
    if OM(y) ≥ OM(x) then x ← y;
    ;
```

3.1 The Drift Maximizing Algorithm

We regard as drift maximizing algorithm the algorithm summarized in Algorithm 3. We denote this algorithm by A^*. A^* starts by querying a uniform solution $x \in \{0,1\}$. In each iteration of the main loop the algorithm generates a new solution y from x by flipping exactly $R(\textsc{Om}(x))$ bits in x, where $R : [0..n] \to [0..n]$ is a function that assigns to each fitness value the number of bits that should be flipped in a search point of this quality. We choose R such that the expected progress (*drift*) is maximized. When there is more than one value maximizing the drift, R chooses the smallest among these. That is,

$$R(\textsc{Om}(x)) := \min\left\{\arg\max E_{y \leftarrow \text{flip}_r(x)}\big(\max\{\textsc{Om}(y) - \textsc{Om}(x), 0\}\big) \mid r \in [0..n]\right\}. \tag{3}$$

In this definition, we make use of the fact that the expected progress $E_{y \leftarrow \text{flip}_r(x)}\big(\max\{\textsc{Om}(y) - \textsc{Om}(x), 0\}\big)$ depends only on the fitness of x but not on its structure. This is due to the symmetry of the function Om. The offspring y replaces its parent x if and only if $\textsc{Om}(y) \geq \textsc{Om}(x)$.

Note here that the function R is deterministic, i.e., we only make use of a unary unbiased mutation operator which deterministically depends on the fitness of the current-best solution. Note further that the search point kept in the memory of algorithm A^* is always a best-so-far solution. A^* can be seen as an RLS-variant with fitness-dependent mutation strength.

3.2 Main Result and Proof Strategy

The main result of this entire section is the following statement, which says that the expected runtime of A^* cannot be much worse than the unary unbiased black-box complexity of OneMax.

Theorem 11. *Let A be a unary unbiased black-box algorithm. Denote by $T(A)$ its runtime on* OneMax *and by $T(A^*)$ the runtime of A^*. Then $E(T(A)) \geq E(T(A^*)) - O(n^{2/3}\ln^9(n))$.*

For the proof of Theorem 11 we derive in Section 3.3 a lower bound for the expected runtime of any unary unbiased algorithm, cf. Theorem 14. We then prove in Section 3.4 an upper bound for the expected runtime of A^*, cf. Theorem 15. For both statements, we use the variable drift theorems presented in Section 2.2. We therefore need to define suitable drift functions which bound from below the expected progress that can be made by algorithm A^* and from above the maximal expected progress that any unary unbiased algorithm can make at every step of the optimization process. This is the purpose of the remainder of this subsection.

3.2.1 The Distance Function

Before we define the drift functions, we first note that in order to maximize the function Om, an optimal algorithm may choose to first minimize the function, and to then flip all bits at once to obtain the optimal Om-solution. Instead of regarding the *maximization of the function* Om, we therefore regard in the following the problem of *minimizing the distance function* d, which

assigns to each string x the value $d(x) := \min\{n - \textsc{Om}(x), \textsc{Om}(x)\}$. The black-box complexities of both problems are almost identical, as the following lemma shows.

Lemma 12. *The unary unbiased black-box complexities of maximizing* OneMax *is at least as large as that of minimizing d and it is larger by at most one.*

Proof. For the first statement, it suffices to observe that we can simulate the optimization of d when OneMax-values are available. The second statement follows from the already mentioned fact that once we have found a string x of distance value $d(x) = 0$, then either x or its bit-wise complement $\bar{x}$ has maximal OneMax-value. □

3.2.2 Drift Expressions

For the definition of the drift functions used in the proofs of Theorems 14 and 15, we use the following notation. For a unary unbiased algorithm A we denote by $(x(0), x(1), \dots, x(t))$ the sequence of the first $t+1$ search points evaluated by A. For every such sequence of search points, we abbreviate by

$$X_t := \min\{d(x(i)) | i \in [0..t]\}$$

the quality of a best-so-far solution with respect to the distance function d. Note that for all $t \geq 0$ it holds that $X_t \geq X_{t+1}$, i.e., the sequence $(X_t)_{t \geq 0}$ is monotonically decreasing in t. For each $k \in [0..n]$ let $\mathcal{H}(t,k)$ be the collection of all sequences $(x(0), x(1), \dots, x(t))$ of search points for which $X_t(x(0), x(1), \dots, x(t)) = k$. Abusing notation, we write $x \in ((x(0), x(1), \dots, x(t)))$ when $x = x(i)$ for some $i \in [0..t]$.

Denoting by $\mathcal{U}$ the set of all unary unbiased operators acting on $\{0,1\}^n$, the maximal possible drift that can be achieved by a unary unbiased variation operator when the best-so-far distance is equal to $k \in [0..n]$ is equal to

$$\hat{h}(k) \;:=\; \max\left\{\max\left\{k - E(\min\{k, d(\tau(x))\}) \mid \tau \in \mathcal{U}, x \in H\right\} \mid t \in \mathbb{N}, H \in \mathcal{H}(t,k)\right\}.$$

Using Lemma 1 it is not difficult to show that

$$\hat{h}(k) \;=\; \max\left\{k - E(\min\{k, d(\mathrm{flip}_r(x))\}) \mid r \in [0..n], x \in \{0,1\}^n \text{ with } d(x) \geq k\right\}. \tag{4}$$

To obtain a monotonically increasing function (as required by Theorem 10), we set

$$h(k) := \max\{\hat{h}(i) \mid i \in [0..k]\}. \tag{5}$$

Note that $h(k) \geq \hat{h}(k)$ for all $k \in [0..n]$.

Finally, we set

$$\tilde{h}(k) \;:=\; \max\{E(\max\{\textsc{Om}(\mathrm{flip}_r(x)) - \textsc{Om}(x), 0\}) \mid x \in \{0,1\}^n \text{ with } \textsc{Om}(x) = n-k, r \in [0..n]\}, \tag{6}$$

the maximal Om-drift that can be obtained from a search point whose Om-value is exactly equal to $n-k$. We certainly have $\tilde{h}(k) \leq h(k)$ for all $k \leq n/2$. However, we will show in Section 3.4.1 that for all values $k \leq n/2 - n^{0.6}$ the difference between the functions h and $\tilde{h}$ is small, showing that we can approximate the maximal drift h by mutating a best-so-far solution. This will be the key step in proving the upper bound for the expected runtime of algorithm A^*. We also notice that $\tilde{h}(k) \geq \max\{\textsc{Om}(\bar{x}) - \textsc{Om}(x), 0\} \geq 2k - n$. Therefore for any $x \in \{0,1\}^n$ with $\textsc{Om}(x) = n - k$ we have $\textsc{Om}(x) + \tilde{h}(k) \geq \max\{\textsc{Om}(x), \textsc{Om}(\bar{x})\} \geq n/2$, so that A^* very quickly has a search point of function value $\textsc{Om}(x) \geq n/2$. Informally, the interesting part of the runtime analysis for A^* is therefore the fitness increase from a value around $n/2$ to n.

3.3 A Lower Bound for all Unary Unbiased Algorithms

Before proving the lower bound, we first introduce the following lemma arguing that the probability to make a large fitness gain is bounded by a small probability, as required by the condition to apply Theorem 10.

Lemma 13. *There exists an $n_0 \in \mathbb{N}$ such that, for all $n \geq n_0$, for all $r \in [0..n]$, and for all $x \in \{0,1\}^n$, it holds that*

$$\Pr\left(d\left(\mathrm{flip}_r(x)\right) \geq \tilde{c}\left(d(x)\right)\right) \geq 1 - n^{-4/3}\ln^7(n), \tag{7}$$

where

$$\tilde{c} : [n] \to [0..n], i \mapsto \tilde{c}(i) := \begin{cases} i - \sqrt{n}\ln n & \text{for } i \geq n/6, \\ i - \ln^2(n) & \text{for } n^{1/3} \leq i < n/6, \\ i - 1 & \text{for } i < n^{1/3}. \end{cases}$$

Proof. Set $p := n^{-4/3}\ln^7(n)$. To show (7), we first note that we can assume without loss of generality that $\textsc{Om}(x) \geq n/2$. This is due to the symmetry of the distance function. In addition, we can assume that $0 < r \leq n/2$, because $d(\mathrm{flip}_r(x))$ and $d(\mathrm{flip}_{n-r}(x))$ are identically distributed.

We make a case distinction according to the size of $d := d(x)$. For all different cases we note that the event $d(\mathrm{flip}_r(x)) < d$ happens in two cases, namely if $\textsc{Om}(\mathrm{flip}_r(x)) > \textsc{Om}(x)$ or if $\textsc{Om}(\mathrm{flip}_r(x)) < n - \textsc{Om}(x)$. We denote by Z the number of good flips in an application of flip_r to x; i.e., the number of bits flipping from 0 to 1. Z follows a hypergeometric distribution and $E(Z) = rd(x)/n$, as discussed in Remark 3.

Case 1: $d(x) \geq n/6$. We first regard the case that $d := d(x) \geq n/6$. The Chernoff bound (e.g., the variants presented in Theorems 1.11 and 1.17 in [Doe11]) applied to Z show that, for all $\lambda > 0$, $\Pr(Z > E(Z) + \lambda) \leq \exp(-2\lambda^2/n)$ and $\Pr(Z < E(Z) - \lambda) \leq \exp(-2\lambda^2/n)$. Using that $\textsc{Om}(\mathrm{flip}_r(x)) = \textsc{Om}(x) + 2Z - r$ and that $n - \textsc{Om}(x) \leq E(\textsc{Om}(\mathrm{flip}_r(x))) \leq \textsc{Om}(x)$, we obtain that

$$\begin{aligned} \Pr\left(\textsc{Om}(\mathrm{flip}_r(x)) < n - \textsc{Om}(x) - 2\lambda\right) &\leq \Pr\left(\textsc{Om}(\mathrm{flip}_r(x)) < E(\textsc{Om}(\mathrm{flip}_r(x))) - 2\lambda\right) \\ &= \Pr\left(\textsc{Om}(x) + 2Z - r < \textsc{Om}(x) + E(2Z) - r - 2\lambda\right) \\ &= \Pr\left(2Z < E(2Z) - 2\lambda\right) \leq \exp(-2\lambda^2/n), \end{aligned} \tag{8}$$

and that

$$\begin{aligned} \Pr\left(\textsc{Om}(\mathrm{flip}_r(x)) > \textsc{Om}(x) + 2\lambda\right) &\leq \Pr\left(\textsc{Om}(\mathrm{flip}_r(x)) > E(\textsc{Om}(\mathrm{flip}_r(x))) + 2\lambda\right) \\ &= \Pr\left(2Z > E(2Z) + 2\lambda\right) \leq \exp(-2\lambda^2/n). \end{aligned} \tag{9}$$

From these two inequalities we conclude that

$$\Pr(d(\mathrm{flip}_r(x)) < \tilde{c}(d(x)) = \Pr(d(\mathrm{flip}_r(x)) < d - \sqrt{n}\ln n) \leq 2\exp(-\ln^2(n)/2) < p$$

for all $d \geq n/6$.

Case 2: $n^{1/3} \leq d(x) < n/6$. By our assumptions $r \leq n/2$ and $\textsc{Om}(x) \geq n/2$ we obtain from Remark 3 that $E(\textsc{Om}(\mathrm{flip}_r(x))) = (1 - \frac{2r}{n})\textsc{Om}(x) + r \geq (1 - \frac{2r}{n})\frac{n}{2} + r = n/2$. Using Chernoff bounds (e.g., Theorems 1.11 and 1.17 in [Doe11]) we thus get $\Pr(\textsc{Om}(\mathrm{flip}_r(x)) < n/6) \leq \exp(-2(n/2 - n/6)^2/n) = o(p)$.

Consider the event $\textsc{Om}(\mathrm{flip}_r(x)) \geq \textsc{Om}(x) + \ln^2(n)$. It intrinsically requires that $r \geq \ln^2(n)$. We apply the Chernoff bound from Corollary 1.10 (b) in [Doe11] to Z and obtain that

$$\Pr\left(\textsc{Om}(\mathrm{flip}_r(x)) > \textsc{Om}(x)\right) = \Pr\left(Z > \frac{r}{2}\right) = \Pr\left(Z > \frac{n}{2d}E(Z)\right) \leq \left(\frac{2de}{n}\right)^{r/2} \tag{10}$$

$$\leq \left(\frac{e}{3}\right)^{r/2} = O(n^{-\Omega(\ln(n))}) = o(p).$$

Therefore

$$\begin{aligned}\Pr(d(\text{flip}_r(x)) < \tilde{c}(d(x)) &= \Pr(d(\text{flip}_r(x)) < d - \ln^2 n) \\ &\leq \Pr(\textsc{Om}(\text{flip}_r(x)) < n/6) + \Pr\left(\textsc{Om}(\text{flip}_r(x)) > \textsc{Om}(x)\right) = o(p)\end{aligned}$$

for all $n^{1/3} \leq d < n/6$.

Case 3: $d(x) < n^{1/3}$. We finally consider the case $d < n^{1/3}$. Applying this condition to the first line of Equation (10), we obtain $\Pr(\textsc{Om}(\text{flip}_r(x)) > \textsc{Om}(x)) \leq \Theta(n^{-4/3}) = o(p)$ for all $r \geq 4$. For $r < 4$ we consider the operators separately and observe that

$$\Pr\left(\textsc{Om}(\text{flip}_3(x)) > \textsc{Om}(x)\right) = \frac{\binom{d}{2}\binom{n-d}{1} + \binom{d}{3}}{\binom{n}{3}} = O(n^{-4/3}) = o(p)$$

and

$$\Pr\left(\textsc{Om}(\text{flip}_2(x)) > \textsc{Om}(x)\right) = \frac{\binom{d}{2}}{\binom{n}{2}} = O(n^{-4/3}) = o(p).$$

Thus, altogether, we obtain that

$$\begin{aligned}\Pr(d(\text{flip}_r(x)) < \tilde{c}(d(x)) &= \Pr(d(\text{flip}_r(x)) < d - 1) \\ &\leq \Pr(\textsc{Om}(\text{flip}_r(x)) < n/6) + \Pr\left(\textsc{Om}(\text{flip}_r(x)) > \textsc{Om}(x)\right) = o(p)\end{aligned}$$

for all $r \geq 2$ and $d < n^{1/3}$. Needless to say that $\Pr(d(\text{flip}_1(x)) < \tilde{c}(d(x)) = 0$. This proves Equation (7). □

Using Lemma 13 and Theorem 10, we are now ready to show the following lower bound.

Theorem 14. *Let $s := n/2 - n^{0.6}$. The expected runtime of any unary unbiased algorithm A on the* OneMax *problem satisfies*

$$E\left(T_A\right) \geq \sum_{x=1}^{s} \frac{1}{h(x)} - \Theta(n^{2/3}\ln^9(n)).$$

Proof. For convenience, we assume that n is sufficiently large. Let A be a unary unbiased algorithm. We recall that by $(x(0), x(1), \dots)$ we denote the sequence of search points evaluated by A and by $X_t = \min\{d(x(i)) \mid i \in [0..t]\}$ the best-so-far distance after first t iterations of the main loop. Let $T := \min\{t \in \mathbb{N} \mid X_t = 0\}$. Then $T \leq T_A := \min\{t \in \mathbb{N} \mid \textsc{Om}(x(t)) = n\}$. We prove a lower bound on T.

Every unary unbiased black-box algorithm has to create its first search point uniformly at random. This initial search point x_0 has expected fitness $E(\textsc{Om}(x_0)) = n/2$. By Chernoff's bound (we can use, for example, the variant presented in Theorem 1.11 in [Doe11]) it furthermore holds that $\Pr(X_0 < s) = \Pr(|\textsc{Om}(x_0) - n/2| > n^{0.6}) \leq 2\exp(-2n^{0.2})$. This probability is small enough such that even optimistically assuming $T_A = 0$ whenever $X_0 < s$, the contribution of such events affect the lower bound by a term of $O(n^{2/3}\ln^9(n))$. We can therefore safely assume that $X_0 \geq s$.

For all $i \in [n]$ let $c(i) := \min\{\tilde{c}(j) \mid j \geq i\}$, where $\tilde{c}$ is the function defined in Lemma 13. For all $r \in [0..n]$, $x \in \{0,1\}^n$, and all distance levels $d' \leq d(x)$ it holds that

$$c(d') \leq \tilde{c}(d(x)) \text{ and } \Pr\left(d(\text{flip}_r(x)) > c(d')\right) \geq \Pr\left(d(\text{flip}_r(x)) > \tilde{c}(d(x))\right) \geq 1 - p.$$

By Lemma 1, this statement can be extended to arbitrary unary unbiased variation operators. Therefore,

$$\Pr\left(X_{t+1} > c(X_t)\right) \geq 1 - p \text{ for all } t \in \mathbb{N}.$$

We apply Theorem 10 to c and h. We first compute μ as in Theorem 10. By definition, $\mu(i) = \max\{x \mid c(x) \leq i\} = \max\{x \mid \min\{\tilde{c}(y) \mid y \geq x\} \leq i\} = \max\{x \mid \tilde{c}(x) \leq i\}$, giving

$$\mu(i) := \begin{cases} i+1 & \text{for } i < n^{1/3} - \ln^2(n), \\ i + \ln^2(n) & \text{for } n^{1/3} - \ln^2(n) \leq i < n/6 - \sqrt{n}\ln n, \\ i + \sqrt{n}\ln n & \text{for } n/6 - \sqrt{n}\ln n \leq i < n/2 - \sqrt{n}\ln n, \\ \lfloor n/2 \rfloor & \text{for } n/2 - \sqrt{n}\ln n \leq i \leq n/2. \end{cases}$$

According to Theorem 10, we can thus bound $E(T \mid X_0)$ by

$$E\left(T \mid X_0\right) \geq g(X_0) - \frac{g^2(X_0)p}{1 + g(X_0)p} \text{ with } g(x) = \sum_{i=1}^{x-1} \frac{1}{h(\mu(x))}.$$

Since $h(\mu(x)) \geq \tilde{h}(\mu(x)) \geq E(\max\{\textsc{Om}(\text{flip}_1(x)) - \textsc{Om}(x), 0\} \mid x \in \{0,1\}^n$ with $\textsc{Om}(x) = n - \mu(x)) = \mu(x)/n \geq x/n$, we obtain $g(X_0) \leq \sum_{i=1}^{X_0-1} \frac{n}{i} = O(n\ln(n))$. Therefore $\frac{g^2(X_0)p}{1+g(X_0)p} = O(n^{2/3}\ln^9(n))$ for $p = n^{-4/3}\ln^7(n)$. Using the monotonicity of h and the fact that all summands are positive, we estimate $g(X_0)$ by

$$\begin{aligned} \sum_{x=0}^{X_0-1} \frac{1}{h(\mu(x))} &\geq \sum_{x=1}^{n^{1/3}-\ln^2(n)} \frac{1}{h(x)} + \sum_{x=n^{1/3}}^{n/6-\sqrt{n}\ln n+\ln^2(n)} \frac{1}{h(x)} + \sum_{x=n/6}^{X_0} \frac{1}{h(x)} \\ &\geq \sum_{x=1}^{X_0} \frac{1}{h(x)} - \frac{\ln^2(n)}{h(n^{1/3} - \ln^2(n))} - \frac{\sqrt{n}\ln n - \ln^2(n)}{h(n/6 - \sqrt{n}\ln n + \ln^2(n))} \\ &\geq \sum_{x=1}^{X_0} \frac{1}{h(x)} - \Theta(n^{2/3}\ln^2(n)). \end{aligned}$$

Therefore we obtain $E(T_A) \geq E(T \mid X_0 \geq s) - O(n^{2/3}\ln^9(n)) \geq \sum_{x=1}^{s} \frac{1}{h(x)} - \Theta(n^{2/3}\ln^9(n))$. □

3.4 Upper Bound for the Drift Maximizer

The lower bound in Theorem 14 also holds for drift-maximizer A^* described in the beginning of this section. We next show that A^* achieves this runtime bound apart from the lower order term $\Theta(n^{2/3}\ln^9(n))$.

Theorem 15. *Let $s := n/2 - n^{0.6}$. The expected runtime of algorithm A^* on* OneMax *satisfies*

$$E\left(T_{A^*}\right) \leq \sum_{x=1}^{s} \frac{1}{h(x)} + \Theta(n^{0.6}).$$

From the variable drift theorem, Theorem 7, we easily get $\sum_{x=1}^{s} \frac{1}{\tilde{h}(x)}$ as upper bound for the expected runtime of A^*. We therefore need to show that the difference between $h(X_t)$ and $\tilde{h}(X_t)$ is small. This is the purpose of the next subsection.

3.4.1 Maximizing Drift by Mutating a Best-So-Far Solution

As mentioned above, we show that the expected drift of any unary unbiased algorithm cannot be significantly better than that of A^*. The main result of this subsection is the following lemma.

Lemma 16. *For sufficiently sufficiently large n and $k \leq n/2 - n^{0.6}$ it holds that $0 \leq h(k) - \tilde{h}(k) \leq n \exp(-\Omega(n^{0.2}))$.*

We start our proof of Lemma 16 by observing that the expected OM-value of the search point obtained from mutating and selecting the best of parent and offspring is strictly increasing with the quality of the parent. The proof is by induction. The base case is covered by the following lemma.

Lemma 17. *Let $x, y \in \{0,1\}^n$ with $\textsc{Om}(y) = \textsc{Om}(x) + 1 \geq n/2$ and let $r \in [0..n/2]$. For all $t \geq 1$ it holds that*

$$\Pr\left(\textsc{Om}(\text{flip}_r(x)) = \textsc{Om}(y) + t\right) \leq \Pr\left(\textsc{Om}(\text{flip}_{r-1}(y)) = \textsc{Om}(y) + t\right). \tag{11}$$

Proof. We first notice that for $t > r - 1$ both two probabilities are zero. We can therefore assume that $1 \leq t \leq r - 1$. Let $d := n - \textsc{Om}(y)$, and let i be the number of zeros in y that flip_{r-1} flips from zero to one. Then there are $r - 1 - i$ ones that flip to zero. We thus have $\textsc{Om}(\text{flip}_{r-1}(y)) = \textsc{Om}(y) + t$ if and only if $i - (r - 1 - i) = t$; i.e., if and only if $i = (t + r - 1)/2$. By the same reasoning $\textsc{Om}(\text{flip}_r(x)) = \textsc{Om}(y) + t$ if and only if $i' - (r - i') = t + 1$ for i' being the number of zeros flipped by flip_r. This implies $i' = (r - 1 + t)/2 = i + 1$. We thus obtain

$$\Pr\left(\textsc{Om}(\text{flip}_{r-1}(y)) = \textsc{Om}(y) + t\right) = \frac{\binom{d}{i}\binom{n-d}{r-1-i}}{\binom{n}{r-1}}$$

$$\Pr\left(\textsc{Om}(\text{flip}_r(x)) = \textsc{Om}(y) + t\right) = \frac{\binom{d+1}{i+1}\binom{n-d-1}{r-(i+1)}}{\binom{n}{r}}$$

To show equation (11), we abbreviate $j := r - 1 - i$ and use the facts that $\binom{d+1}{i+1} = \binom{d}{i}\frac{d+1}{i+1}$, $\binom{n-d}{j} = \binom{n-d-1}{j}\frac{n-d}{n-d-j}$, and $\binom{n}{i+j+1} = \binom{n}{i+j}\frac{n-i-j}{i+j+1}$ to obtain that

$$\frac{\frac{\binom{d+1}{i+1}\binom{n-d-1}{r-(i+1)}}{\binom{n}{r}}}{\frac{\binom{d}{i}\binom{n-d}{r-1-i}}{\binom{n}{r-1}}} = \frac{(d+1)(n-d-j)(1+i+j)}{(i+1)(n-d)(n-i-j)}.$$

We aim to show the above ratio less or equal to 1. To this end, we compute the difference between the numerator and the denominator, and obtain $(d+1)(n-d-j)(1+i+j) - (i+1)(n-d)(n-i-j) = (1+i+j+d-n)(n+in-(1+i+j)d)-j) = (r+d-n)(n+in-rd-j)$. The first factor in this expression is negative, since both $r \leq n/2$ and $d \leq n/2$. The second factor is positive, because $n > j$, $i > r/2$ and $d \leq n/2$ implying that $in - rd > (r/2)n - r(n/2) \geq 0$. □

We now regard the case that the same number r of bits are flipped in the two strings x and y.

Lemma 18. *Let $x, y \in \{0,1\}^n$ with $\textsc{Om}(y) = \textsc{Om}(x) + 1$ and let $r \in [0..n]$. It holds that*

$$E\left(\max\{\textsc{Om}(\text{flip}_r(x)) - \textsc{Om}(x), 0\}\right) \geq E\left(\max\{\textsc{Om}(\text{flip}_r(y)) - \textsc{Om}(y), 0\}\right). \tag{12}$$

Proof. Since permutation on bit-positions does not affect the analysis, we assume that x and y are of the following form.

$$x = \underbrace{11\cdots11}_{\frac{n}{2}+\delta}\underbrace{00\cdots00}_{\frac{n}{2}-\delta-1}0 \text{ and } y = \underbrace{11\cdots11}_{\frac{n}{2}+\delta}\underbrace{00\cdots00}_{\frac{n}{2}-\delta-1}1. \tag{13}$$

For any r-sized subset S of $[n]$ let x_S (y_S) denote the offspring of x (y) in which the r positions in S are flipped. Then x_S and y_S differ only in the last bit and we have $\textsc{Om}(x_S)-\textsc{Om}(y_S) \in \{-1,1\}$ for all S. Therefore

$$\max\{\textsc{Om}(\text{flip}_r(x)), \textsc{Om}(x)\} - \max\{\textsc{Om}(\text{flip}_r(y)), \textsc{Om}(y)\} \geq -1 \text{ for all } S$$

Using that $\textsc{Om}(y) - \textsc{Om}(x) = 1$ we obtain

$$\begin{aligned} & E\left(\max\{\textsc{Om}(\text{flip}_r(x)) - \textsc{Om}(x), 0\}\right) - E\left(\max\{\textsc{Om}(\text{flip}_r(y)) - \textsc{Om}(y), 0\}\right) \\ = \; & E\left(\max\{\textsc{Om}(\text{flip}_r(x)), \textsc{Om}(x)\} - \textsc{Om}(x)\right) - E\left(\max\{\textsc{Om}(\text{flip}_r(y)), \textsc{Om}(y)\} - \textsc{Om}(y)\right) \\ = \; & E\left(\max\{\textsc{Om}(\text{flip}_r(x)), \textsc{Om}(x)\} - \max\{\textsc{Om}(\text{flip}_r(y)), \textsc{Om}(y)\}\right) + 1 \geq 0. \end{aligned}$$

□

We now extend the last two lemmas to the case $\textsc{Om}(y) - \textsc{Om}(x) > 1$.

Corollary 19. *Let $x, y \in \{0,1\}^n$ with $\textsc{Om}(y) > \textsc{Om}(x)$.*

1. *if $\textsc{Om}(x) \geq n/2$, $r \in [0..n/2]$, and $r' = \max\{r - (\textsc{Om}(y) - \textsc{Om}(x)), 0\}$. Then for all $t \in \mathbb{N}_{\geq 1}$ we have*

$$\Pr\left(\textsc{Om}(\text{flip}_r(x)) = \textsc{Om}(y) + t\right) \leq \Pr\left(\textsc{Om}(\text{flip}_{r'}(y)) = \textsc{Om}(y) + t\right). \tag{14}$$

2. *It also holds that*

$$\tilde{h}(\textsc{Om}(x)) \geq \tilde{h}(\textsc{Om}(y)).$$

Proof. To see the first statement, we first regard the case that $r' = 0$. In this case, we have $r \leq \textsc{Om}(y) - \textsc{Om}(x)$, so that the probability that $\textsc{Om}(\text{flip}_r(x)) = \textsc{Om}(y) + t$ is zero for $t > 0$. For $t = 0$ the statement trivially holds, since the right-hand side of (14) is equal to one. For $r' > 0$ the first statement follows from Lemma 17 and an induction over $\textsc{Om}(y) - \textsc{Om}(x)$.

To prove the second statement we first assume that $\textsc{Om}(y) - \textsc{Om}(x) = 1$. Let r be the value that maximizes $E\left(\max\{\textsc{Om}(\text{flip}_r(y)) - \textsc{Om}(y), 0\}\right)$. Using Lemma 18 we obtain

$$\begin{aligned} \tilde{h}(\textsc{Om}(y)) &= E\left(\max\{\textsc{Om}(\text{flip}_r(y)) - \textsc{Om}(y), 0\}\right) \\ &\leq E\left(\max\{\textsc{Om}(\text{flip}_r(x)) - \textsc{Om}(x), 0\}\right) \leq \tilde{h}(\textsc{Om}(x)). \end{aligned}$$

The general statement now follows by induction over $\textsc{Om}(y) - \textsc{Om}(x)$.

□

We are now ready to prove the main result of this subsection, Lemma 16.

Proof of Lemma 16. Let $k \leq n/2 - n^{0.6}$, x a search point with $d(x) \geq k$ and let $r \in [0..n]$. Since all random variables $d(\text{flip}_r(x))$, $d(\text{flip}_{n-r}(x))$, $d(\text{flip}_r(\bar{x}))$, and $d(\text{flip}_{n-r}(\bar{x}))$ are identically distributed, we can assume without loss of generality that $\textsc{Om}(x) \geq n/2$ and that $r \leq n/2$.

Using this and the observations made in Remark 3, we easily see that

$$E(\textsc{Om}(\text{flip}_r(x))) = \textsc{Om}(x) - r\tfrac{\textsc{Om}(x)}{n} + r\tfrac{n-\textsc{Om}(x)}{n} = \textsc{Om}(x)(1 - r/n) + (n - \textsc{Om}(x))r/n \geq n/2.$$

This shows that $E(\textsc{Om}(\text{flip}_r(x))) - n^{0.6} \geq n/2 - n^{0.6} \geq k$. Together with a Chernoff bound applied to $\textsc{Om}(\text{flip}_r(x))$ we thus obtain

$$\Pr\left(\textsc{Om}(\text{flip}_r(x)) < k\right) \leq \Pr\left(\textsc{Om}(\text{flip}_r(x)) < E(\textsc{Om}(\text{flip}_r(x))) - n^{0.6}\right) = \exp(-\Omega(n^{0.2})). \quad (15)$$

We first aim at bounding $\hat{h}(k)$. To this end, we use the estimate 15 to obtain

$$\begin{aligned}
&E\left(k - \min\{k, d(\text{flip}_r(x))\}\right) \\
&\quad = E\left(\textsc{Om}(\text{flip}_r(x)) - (n-k) \mid \textsc{Om}(\text{flip}_r(x)) > n-k\right) \Pr\left(\textsc{Om}(\text{flip}_r(x)) > n-k\right) \\
&\qquad + E\left(k - \textsc{Om}(\text{flip}_r(x)) \mid \textsc{Om}(\text{flip}_r(x)) < k\right) \Pr\left(\textsc{Om}(\text{flip}_r(x)) < k\right) \\
&\quad \leq E\left(\max\{\textsc{Om}(\text{flip}_r(x)) - (n-k), 0\}\right) + n\exp(-\Omega(n^{0.2})).
\end{aligned}$$

Let x' be a search point with $\textsc{Om}(x') = n-k$, and let $r' = \max\{r - (d(x) - d(x')), 0\}$. According to Corollary 19 it holds for all $i \in \mathbb{N}$ that

$$\Pr(\textsc{Om}(\text{flip}_r(x)) = n-k+i) \leq \Pr(\textsc{Om}(\text{flip}_{r'}(x')) = n-k+i).$$

Using that $E\left(\max\{\textsc{Om}(\text{flip}_r(x)) - (n-k), 0\}\right) = \sum_{i \geq 1} i \Pr(\textsc{Om}(\text{flip}_r(x)) = n-k+i)$ we obtain

$$E\left(k - \min\{k, d(\text{flip}_r(x))\}\right) \leq E\left(\max\{\textsc{Om}(\text{flip}_{r'}(x')) - (n-k), 0\}\right) + n\exp(-\Omega(n^{0.2})).$$

Referring to the definition of $\hat{h}$ and $\tilde{h}$ in equations (4) and (6), and using the symmetries mentioned in the beginning of this proof, we bound

$$\begin{aligned}
\hat{h}(k) &= \max\{k - E\left(\min\{k, d(\text{flip}_r(x))\}\right) \mid r \in [0..n], x \in \{0,1\}^n \text{ with } d(x) \geq k\} \\
&= \max\{k - E\left(\min\{k, d(\text{flip}_r(x))\}\right) \mid r \in [0..n/2], x \in \{0,1\}^n \text{ with } n/2 \leq \textsc{Om}(x) \leq n-k\} \\
&\leq \max\{E\left(\max\{\textsc{Om}(\text{flip}_{r'}(x')) - (n-k), 0\}\right) \mid r' \in [0..n/2], x' \in \{0,1\}^n \text{ with } \textsc{Om}(x') = n-k\} \\
&\quad + n\exp(-\Omega(n^{0.2})) \\
&\leq \tilde{h}(k) + n\exp(-\Omega(n^{0.2})).
\end{aligned}$$

According to the definition of $h(k)$ in Equation (5), we obtain

$$h(k) = \max\{\hat{h}(i) \mid i \in [0..k]\} \leq \max\{\tilde{h}(i) \mid i \in [0..k]\} + n\exp(-\Omega(n^{0.2}) = \tilde{h}(k) + n\exp(-\Omega(n^{0.2}),$$

where the last equality uses the monotonicity of $\tilde{h}(k)$ with respect to k shown in Corollary 19. □

As we will see in the next subsection, the $n\exp(-\Omega(n^{0.2}))$ term in this bound accounts for an additive $O(1)$ error in the runtime estimate only.

3.4.2 Proof of Theorem 15

With Lemma 16 at hand, we are now ready to prove Theorem 15.

Proof of Theorem 15. As mentioned above, we easily obtain from Theorem 7 that

$$E\left(T_{A^*} \mid x(0)\right) \leq \sum_{x=1}^{n-\textsc{Om}(x(0))} \frac{1}{\tilde{h}(x)}. \quad (16)$$

According to Lemma 16 it holds that $0 < h(x) - \tilde{h}(x) \leq n\exp(-\Omega(n^{0.2}))$ for $x \leq n/2 - n^{0.6}$. Using again that flipping a single bit on a bit string with x zeros gives an expected progress in the $\textsc{Om}$-value of x/n, we recall that $h(x) \geq \tilde{h}(x) \geq x/n$ for all $x \in [n/2]$. Therefore,

$$\frac{1}{\tilde{h}(x)} - \frac{1}{h(x)} = \frac{h(x) - \tilde{h}(x)}{\tilde{h}(x)h(x)} \leq \frac{n\exp(-\Omega(n^{0.2}))}{(x/n)(x/n)} \leq n^3\exp(-\Omega(n^{0.2})) \text{ for all } 1 \leq x \leq s.$$

Replacing $\tilde{h}(x)$ by $h(x)$ in inequality (16) and pessimistically assuming $\textsc{Om}(x(0)) = 0$ we thus obtain

$$E\left(T_{A^*}\right) \leq \sum_{x=1}^{s}\left(\frac{1}{h(x)} + n^3\exp(-\Omega(n^{0.2}))\right) + \sum_{x=s}^{n}\frac{1}{\tilde{h}(x)} = \sum_{x=1}^{s}\frac{1}{h(x)} + \sum_{x=s}^{n}\frac{1}{\tilde{h}(x)} + O(1).$$

Using that $\tilde{h}(x) \geq x/n$ for all $0 < x \leq n$ and $\tilde{h}(x) \geq 2x - n$ for all $n/2 < x \leq n$, we conclude

$$\sum_{x=s}^{n}\frac{1}{\tilde{h}(x)} \leq \sum_{x=s}^{n/2+n^{0.6}}\frac{n}{x} + \sum_{x=n/2+n^{0.6}}^{n}\frac{1}{2x-n} \leq \frac{2n^{1.6}}{s} + \frac{n/2}{2n^{0.6}} = \Theta(n^{0.6}).$$

□

Theorem 11 follows from Theorems 14 and 15 by observing that $\Theta(n^{0.6}) = o(n^{2/3}\ln^9(n))$.

4 Fitness-Dependent Mutation Strength

In the previous section we have seen that in order to compute the expected runtime of a best possible unary unbiased black-box algorithm for OneMax we can regard the algorithm A^* that maximizes at any point in time the fitness drift. By Theorems 14 and 15 this algorithm cannot be worse (in expectation) than an optimal unary unbiased one by more than an additive $\Theta(n^{2/3}\ln^9(n))$ term.

In this section we give a relatively concise description of A^*, i.e., we compute approximately the number of bits that need to be flipped in order to maximize the fitness drift. Since we are here talking about the drift in the fitness, it will be convenient to denote in this section by $d(x) = n - \textsc{Om}(x)$ the fitness distance to the target. We also denote by

$$R_{\text{opt}}(d,n) := \min\left\{\arg\max E_{y\leftarrow\text{flip}_r(x)}\left(\max\{\textsc{Om}(y) - \textsc{Om}(x), 0\}\right) \mid r \in [0..n], \textsc{Om}(x) = n - d\right\}, \tag{17}$$

the number of bits that need to be flipped in a search point $x \in \{0,1\}^n$ with $\textsc{Om}(x) = n - d$ such that the expected drift $E(\max\{0, \textsc{Om}(\text{flip}_{R_{\text{opt}}(d,n)}(x)) - \textsc{Om}(x)\})$ is maximized (breaking ties by flipping fewer bits).

The exact analysis of R_{opt} is rather tedious, as we will demonstrate below. Luckily, it turns out that we can safely approximate this point-wise drift maximizing function $R_{\text{opt}}(d,n)$ by some a function $\tilde{R}_{\text{opt}} : [0,1] \to [0..n]$ which maps the relative fitness distance $d(x)/n$ to a mutation strength. Since $\tilde{R}_{\text{opt}}$ is much easier to work with, this is the focus of Section 4.2. For the approximation $\tilde{R}_{\text{opt}}$ we make use of the fact that for values of r that are reasonably small compared to the problem dimension n and the current fitness distance $d(x)$, the expected drift $E(\max\{0, \textsc{Om}(\text{flip}_{R_{\text{opt}}(d,n)}(x)) - \textsc{Om}(x)\})$ is almost determined by the relative fitness distance $d(x)/n$. For very small $d(x) = o(n)$ the fitness drift of flipping $R_{\text{opt}}(d/n) = 1$ bit is exactly $d(x)/n$, without any estimation error. We will also see in Section 4.1 that it suffice to regard constant values r.

Once the approximation of the function R_{opt} by $\tilde{R}_{\text{opt}}$ is established, we demonstrate in Section 4.3 a few properties of these two functions that will be useful in our subsequent computations; in particular for the numerical approximation of $\tilde{R}_{\text{opt}}$, which is carried out in Section 5.2. Most importantly, we shall see that $\tilde{R}_{\text{opt}}$ is monotone, i.e., the number of bits to flip in order to maximize the approximated point-wise drift decreases with increasing fitness. We also show that both R_{opt} and $\tilde{R}_{\text{opt}}$ take only odd values, implying that flipping an even number of bits is suboptimal in all stages of the optimization process.

To ease the computation of $\tilde{R}_{\text{opt}}$, we analyze in detail the mutation rate for search points $x(t)$ with fitness distance $X_t \leq (1/2 - \varepsilon)n$, where the constant ε satisfies $0 < \varepsilon < 1/2$. Notice that by selecting between parent and its offspring at the end of each iteration in Algorithm 3 we have $\textsc{Om}(x(t)) = n - X_t$. For the remaining fitness distances, we simply take $\tilde{R}_{\text{opt}}(1/2 - \varepsilon)$ for all $(1/2 - \varepsilon)n < X_t \leq n/2$ and n for all $X_t > n/2$. A detailed definition of $\tilde{R}_{\text{opt},\varepsilon}$ is provided in equation (22). We will prove in Theorem 34 that our adhoc definition of $\tilde{R}_{\text{opt},\varepsilon}(p)$ for $p \geq 1/2 - \varepsilon$ only causes an error term of $O(\varepsilon n)$ in the runtime.

4.1 The Exact Fitness Drift $B(n, d, r)$

In this subsection, we compute the exact fitness gain obtained from flipping r bits. We shall then argue that once we have a fitness of at least $(\frac{1}{2} + \varepsilon)n$ for some constant ε, the maximal fitness drift stems from flipping some constant number of bits.

Let x be a binary string of length n with fitness distance d, that is, with OneMax-value $n - d$. By the symmetry of the OneMax function, the expected progress of flipping r bits in x does not depend on the structure of x but only on its fitness. We can therefore define the expected fitness gain from flipping r random bits in x by

$$B(n, d, r) := E\left(\max\{0, d - d(\text{flip}_r(x))\} \mid \textsc{Om}(x) = n - d\right).$$

To compute $B(n, d, r)$ arithmetically, let us assume that $i \in [0..r]$ is the number of bits flipped from 0 to 1. Then $r - i$ bits have flipped in the opposite direction from 1 to 0, resulting in a progress of $i - (r - i)$. This progress is positive if and only if $i > r - i$, i.e., if and only if $i > r/2$. The probability for i bits flipping in the "good" direction is $\binom{d}{i}\binom{n-d}{r-i}/\binom{n}{r}$. We therefore obtain

$$B(n, d, r) = \sum_{i=\lceil r/2 \rceil}^{r} \frac{\binom{d}{i}\binom{n-d}{r-i}(2i - r)}{\binom{n}{r}}.$$

We show that the maximal fitness drift is obtained from flipping a constant number of bits once we have a fitness of at least $(\frac{1}{2} + \varepsilon)n$. The main argument is that flipping a single random bit already gives a better expected fitness gain than flipping many bits, which is due to the fact that when flipping many bits, the strong concentration of the hypergeometric distributions renders it highly unlikely that a fitness gain is obtained at all.

Lemma 20. *Let $0 < \varepsilon < 1/2$ and $\alpha := 2\log(4/(\varepsilon^2(1/2 - \varepsilon))$. Then for all $n \in \mathbb{N}$, $d \leq (1/2 - \varepsilon)n$, and all $r \geq 2\alpha/\varepsilon^2$, we have $B(n, d, r) < B(n, d, 1)/2$.*

Proof. Let Z denote the number of "good" flips (i.e., the number of bits flipping from 0 to 1). As discussed in Remark 3, the random variable Z follows a hypergeometric distribution with mean value $E(Z) = dr/n = pr < r/2$, where we abbreviate $p := d/n$. Applying the Chernoff bound presented in Theorem 1.9 (b) in [Doe11] to Z, we obtain

$$\begin{aligned} \Pr\left(Z > r/2\right) = \Pr\left(Z > E(Z)/(2p)\right) &\leq \left(\frac{e^{1/(2p)-1}}{(1/(2p))^{1/(2p)}}\right)^{E(Z)} \\ = \left((2p)^{1/(2p)} e^{1/(2p)-1}\right)^{rp} &= \left(\frac{(2pe)^{1/(2p)}}{e}\right)^{rp} = \left(\frac{2pe}{e^{2p}}\right)^{r/2}. \end{aligned} \tag{18}$$

We then regard $B(n, d, r)/(B, n, d, 1) \leq r \Pr(Z > r/2)/(d/n) = (r/p)\left(\frac{2pe}{e^{2p}}\right)^{r/2} = r(2e)^{r/2} p^{r/2-1} e^{-pr}$. We notice that for fixed $r \geq 2\alpha/\varepsilon^2$ and $0 < p < 1/2 - \varepsilon$,

$$(p^{r/2-1}e^{-pr})' = (r/2 - 1)p^{r/2-2}e^{-pr} - rp^{r/2-1}e^{-pr} = p^{r/2-2}e^{-pr}(r/2 - 1 - rp)$$

$$\geq p^{r/2-2}e^{-pr}(2\alpha/\varepsilon - 1) > 0,$$

thus it remains to check the statement for $d = (1/2-\varepsilon)n$. Using the Taylor expansion $e^{2\delta} = 1 + 2\delta + 2\delta^2 + (4/3)\delta^3 + O(\delta^4) < 1 + 2\delta + 3\delta^2$ we see that $\lim_{p\to 1/2-\varepsilon} 2pe/e^{2p} = \lim_{\delta\to\varepsilon} 2(1/2-\delta)e/e^{2(1/2-\delta)} = \lim_{\delta\to\varepsilon}(1-2\delta)e^{2\delta} < \lim_{\delta\to\varepsilon}(1-2\delta)(1+2\delta+3\delta^2) = 1-\varepsilon^2-6\varepsilon^3 < 1-\varepsilon^2$. Therefore we obtain $B(n,d,r) \leq r\Pr(Z > r/2) \leq r(2pe/e^{2p})^{r/2} < r(1-\varepsilon^2)^{r/2}$. Using the fact that $(1-\varepsilon^2)^{1/\varepsilon^2} < 1/e$, $r \geq 2\alpha/\varepsilon^2 > 2/\varepsilon^2$, and $r(1-\varepsilon^2)^{r/2}$ monotonically decreases when $r > 2/\varepsilon^2$, we obtain $B(n,d,r) < r\exp(-\alpha) \leq 2\alpha\exp(-\alpha)/\varepsilon^2$. Since $\log(\alpha\exp(-\alpha)) = \log(\alpha)-\alpha < -\alpha/2$, then $B(n,d,r) < 2\alpha\exp(-\alpha)/\varepsilon^2 < 2\exp(-\alpha/2)/\varepsilon^2 = (2/\varepsilon^2)(\varepsilon^2(1/2-\varepsilon)/4) = (1/2-\varepsilon)/2 = B(n,d,1)/2$ for $d = (1/2-\varepsilon)n$. □

4.2 Approximating $B(n,d,r)$ via $A(r, \frac{d}{n}, 1-\frac{d}{n})$

When n and d are large compared to r, the expected progress $B(n,d,r)$ is almost determined by d/n. This inspires the following definition of $A(r,p,q)$ which will have the property that $A(r, \frac{d}{n}, 1-\frac{d}{n})$ is a good approximation of $B(n,d,r)$. The definition for general p and q instead of p and $q = 1-p$ will be useful in the following proofs.

Definition 21. *For all $r \in \mathbb{N}$, $p \in [0,1]$, and $q \in [0,1]$, let*

$$A(r,p,q) := \sum_{i=\lceil r/2\rceil}^{r} \binom{r}{i}(2i-r)p^i q^{r-i}. \tag{19}$$

The following Theorem 22 makes precise how well for $p = d/n$ and $q = 1-p$ the value $A(r,p,q)$ approximates the expected progress $B(n,d,r)$.

Theorem 22. *Let $0 < \varepsilon < 1/2$ and $\alpha = 2\log(4/(\varepsilon^2(1/2-\varepsilon))$ (as in Lemma 20). Then for all $n \in \mathbb{N}$ large enough, all $r < 2\alpha/\varepsilon^2$, and all $2r \leq d \leq (1/2-\varepsilon)n$, we have*

$$\left| A\left(r, \frac{d}{n}, \frac{n-d}{n}\right) - B(n,d,r)\right| < \frac{3r^3}{d}. \tag{20}$$

The first step in the proof of Theorem 22 is the following statement, which compares suitable A-values with B-values. Note that here we profit from the general definition of $A(r,p,q)$ instead of the special case $A(r,p,1-p)$.

Lemma 23. *Let $n \in \mathbb{N}$ be sufficiently large and $1 \leq r \leq d \leq (1/2-\varepsilon)n$ with $0 < \varepsilon < 1/2$. It holds that*

$$A\left(r, \frac{d}{n}, \frac{n-d}{n-r}\right) \geq B(n,d,r) \geq A\left(r, \frac{d-r}{n}, \frac{n-d-r}{n}\right). \tag{21}$$

Proof. For any two positive integers r and $i \leq r$ we abbreviate

$$(r)_i := r(r-1)\dots(r-i+1) = \prod_{j=0}^{i-1}(r-j).$$

With this notation, we can express $B(n,m,r)$ as

$$B(n,d,r) = \sum_{i=\lceil r/2\rceil}^{r} \frac{\binom{d}{i}\binom{n-d}{r-i}(2i-r)}{\binom{n}{r}} = \sum_{i=\lceil r/2\rceil}^{r} \frac{(d)_i(n-d)_{r-i}}{(n)_r}\binom{r}{i}(2i-r).$$

From the elementary fact that for all $\lceil r/2 \rceil \leq i \leq r$, we have $(n-d)_{r-i} \leq (n-d)^{r-i}$ and $(n)_r \geq (n)_i (n-r)^{r-i}$, we obtain

$$\frac{(d)_i (n-d)_{r-i}}{(n)_r} \leq \frac{(d)_i (n-d)_{r-i}}{(n)_i (n-r)^{r-i}} \leq \left(\frac{d}{n}\right)^i \left(\frac{n-d}{n-r}\right)^{r-i}.$$

This shows $B(n,d,r) \leq A\left(r, \frac{d}{n}, \frac{n-d}{n-r}\right)$.

To show the second inequality, we use the fact that for all $i \leq r \leq n$, we have $(n)_r \leq n^r$, $(d)_i \geq (d-r)^i$, and $(n-d)_{r-i} \geq (n-d-r)^{r-i}$. Consequently,

$$\frac{(d)_i (n-d)_{r-i}}{(n)_r} \geq \frac{(d-r)^i (n-d-r)^{r-i}}{n^r} = \left(\frac{d-r}{n}\right)^i \left(\frac{n-d-r}{n}\right)^{r-i},$$

yielding $B(n,d,r) \geq A\left(r, \frac{d-r}{n}, \frac{n-d-r}{n}\right)$. □

With Lemma 23 at hand, we now prove Theorem 22.

Proof of Theorem 22. According to the definition of $A(r,p,q)$ in (19) we have

$$\frac{\partial A(r,p,q)}{\partial q} = \sum_{i=\lceil r/2 \rceil}^{r} (r-i)\binom{r}{i}(2i-r)p^i q^{r-i-1} < \frac{r}{2} \sum_{i=\lceil r/2 \rceil}^{r} \binom{r}{i}(2i-r)p^i q^{r-i-1} = \frac{rA(r,p,q)}{2q} < \frac{r^2}{2q},$$

where we have used in the last step that $A(r,p,q) \leq r$.

Using that $\frac{n-d}{n} > 1/2$ and $0 < \frac{n-d}{n-r} - \frac{n-d}{n} = \frac{n-d}{n-r} \cdot \frac{r}{n} < (1+o(1))\frac{r}{n}$, we bound

$$A\left(r, \frac{d}{n}, \frac{n-d}{n-r}\right) - A\left(r, \frac{d}{n}, \frac{n-d}{n}\right) \leq \frac{r^2}{2(1/2)}\left(\frac{n-d}{n-r} - \frac{n-d}{n}\right) \leq \frac{(1+o(1))r^3}{n}.$$

Similarly we have

$$\frac{\partial A(r,p,q)}{\partial p} = \sum_{i=\lceil r/2 \rceil}^{r} i\binom{r}{i}(2i-r)p^{i-1}q^{r-i} < r \sum_{i=\lceil r/2 \rceil}^{r} \binom{r}{i}(2i-r)p^{i-1}q^{r-i} = \frac{rA(r,p,q)}{p} < \frac{r^2}{p}.$$

Using $d \geq 2r$ we obtain

$$A\left(r, \frac{d}{n}, \frac{n-d-r}{n}\right) - A\left(r, \frac{d-r}{n}, \frac{n-d-r}{n}\right) \leq \frac{r^2}{(d-r)/n} \cdot \frac{r}{n} = \frac{r^3}{d-r} > \frac{2r^3}{d}.$$

Therefore

$$A\left(r, \frac{d}{n}, \frac{n-d}{n-r}\right) - A\left(r, \frac{d-r}{n}, \frac{n-d-r}{n}\right) = \frac{(1+o(1))r^3}{n} + \frac{2r^3}{d} < \frac{3r^3}{d}.$$

By Lemma 23, it suffices to estimate $|A(r, \frac{d}{n}, \frac{n-d}{n}) - B(n,d,r)| < A(r, \frac{d}{n}, \frac{n-d}{n-r}) - A(r, \frac{d-r}{n}, \frac{n-d-r}{n}) < \frac{3r^3}{d}$. □

4.3 Approximate Optimal Number of Bits to Flip

The goal of this section is to approximate the function R_{opt} which tells us how many bits one should flip in order to maximize the point-wise drift. Given Theorem 22 above, it is tempting to assume that the map $p \mapsto \arg\max_{r \in \mathbb{N}} A(r,p,1-p)$ should do. Analogous to Lemma 20 we show in the following lemma that it suffices to regard constant r for the approximated drift.

Lemma 24. *Let $0 < \varepsilon < 1/2$ and $\alpha := 2\log(4/(\varepsilon^2(1/2-\varepsilon))$. For all $n \in \mathbb{N}$ with $d \leq (1/2-\varepsilon)n$ and all $r \geq 2\alpha/\varepsilon^2$, the expected approximated drift $A(r, \frac{d}{n}, \frac{n-d}{n}) < A(1, \frac{d}{n}, \frac{n-d}{n})/2$.*

Proof. Consider the binomial random variable $Z \sim \mathrm{Bin}(r, d/n)$. Let $p = d/n$ then $E(Z) = pr$. Applying the Chernoff bound presented in Theorem 1.9 (b) in [Doe11] to Z, we obtain

$$\Pr\left(Z > r/2\right) = \Pr\left(Z > E(Z)/(2p)\right) \leq \left(\frac{e^{1/(2p)-1}}{(1/(2p))^{1/(2p)}}\right)^{E(Z)} = \left(\frac{2pe}{e^{2p}}\right)^{r/2},$$

which is the same inequality as (18) in Lemma 20. Since $A(r,p,1-p) := \sum_{i=\lceil r/2 \rceil}^{r} \binom{r}{i}(2i-r)p^i(1-p)^{r-i} \leq r\sum_{i=0}^{r} \mathbb{1}_{2i>r}\binom{r}{i}p^i(1-p)^{r-i} = r\Pr(Z > r/2)$ and $A(1,p,1-p) = p$, with the same arguments as in Lemma 20, the statement holds. □

In the remainder of this section we show that flipping a number r of bits that maximizes $A(r, d/n, 1-d/n)$ yields indeed a good approximation of the best possible expected progress. Since in principle there could be more than one r maximizing $A(r,p,1-p)$ for a given relative distance $p \in (0, 1/2)$, we break ties by preferring smaller values of r. For $p \geq 1/2-\varepsilon$, where our reasoning above was not applicable, we do not try to find an optimal number of bits to flip, but rather one that does the job of giving a near-optimal runtime. Since a random initial search point has a fitness close to $n/2$, not too much time is spent in this regime anyway. Consequently, we define, for all $\varepsilon > 0$,

$$\tilde{R}_{\mathrm{opt},\varepsilon}(p) := \begin{cases} \min\left\{\arg\max_{r\in\mathbb{N}} A(r,p,1-p)\right\} \text{ for } 0 < p \leq 1/2-\varepsilon, \\ \tilde{R}_{\mathrm{opt}}\left(1/2-\varepsilon\right) \text{ for } 1/2-\varepsilon < p \leq 1/2, \\ n \text{ for } p > 1/2, \end{cases} \tag{22}$$

According to Lemma 24 the function $\tilde{R}_{\mathrm{opt},\varepsilon}$ is well defined (for all $\varepsilon > 0$).

We prove two important properties of the functions $\tilde{R}_{\mathrm{opt},\varepsilon}$, which are summarized in the following theorem.

Theorem 25. *For all $\varepsilon > 0$ the function $\tilde{R}_{\mathrm{opt},\varepsilon}$ is monotonically increasing with respect to p. For all $d \leq n/2$, $\tilde{R}_{\mathrm{opt},\varepsilon}(d/n)$ and $R_{\mathrm{opt}}(d,n)$ are odd values.*

The proof of the second claim in Theorem 25 will be carried out in Section 4.3.1. It is purely combinatorial. The proof of the monotonicity of $\tilde{R}_{\mathrm{opt},\varepsilon}$, in contrast, is surprisingly technical. It will be carried out in Section 4.3.2.

4.3.1 R_{opt} and $\tilde{R}_{\mathrm{opt}}$ Attain Only Odd Values

One possibly surprising property of the functions R_{opt} and $\tilde{R}_{\mathrm{opt},\varepsilon}$ is that they take only odd values. That is, regardless of how far we are from the optimum, the maximal drift is obtained for an odd number of bit flips. The following two lemmas show this statement for the approximate and the exact drift, respectively.

Lemma 26 (flipping even numbers of bits is sub-optimal, statement for the approximated drift A)**.** *For all $k \in \mathbb{N}$ and $p \in (0,1)$ it holds that $\frac{A(2k,p,1-p)}{2k} = \frac{A(2k+1,p,1-p)}{2k+1}$. Consequently $\tilde{R}_{\mathrm{opt},\varepsilon}(d/n)$ takes odd values for all $d \leq n/2$ and all $\varepsilon > 0$.*

Proof. By definition of the function A in (19) and using the facts that for all $r \in \mathbb{N}$ and all $i \leq r$ we have

$$\binom{r}{i} = \binom{r}{r-i}, \quad \binom{r}{i}i = r\binom{r-1}{i-1}, \text{ and } \binom{r}{i}(r-i) = r\binom{r-1}{i},$$

we easily see that

$$\begin{aligned} A(r,p,q) &= \sum_{i=\lceil r/2\rceil}^{r} \binom{r}{i}(2i-r)p^i q^{r-i} \\ &= \sum_{i=\lceil r/2\rceil}^{r} \left(\binom{r}{i} i - \binom{r}{i}(r-i)\right) p^i q^{r-i} \\ &= r \sum_{i=\lceil r/2\rceil}^{r} \left(\binom{r-1}{i-1} - \binom{r-1}{i}\right) p^i q^{r-i}. \end{aligned} \tag{23}$$

This shows that, for all $k \in \mathbb{N}$,

$$\begin{aligned} \frac{A(2k+1,p,q)}{2k+1} &= \sum_{i=k+1}^{2k+1} \left(\binom{2k}{i-1} - \binom{2k}{i}\right) p^i q^{2k+1-i} \\ &= \sum_{i=k+1}^{2k+1} \left(\binom{2k-1}{i-1} - \binom{2k-1}{i} + \binom{2k-1}{i-2} - \binom{2k-1}{i-1}\right) p^i q^{2k+1-i} \\ &= \sum_{i=k+1}^{2k} \left(\binom{2k-1}{i-1} - \binom{2k-1}{i}\right) p^i q^{2k+1-i} + \sum_{i=k+1}^{2k+1} \left(\binom{2k-1}{i-2} - \binom{2k-1}{i-1}\right) p^i q^{2k+1-i} \\ &= \sum_{i=k+1}^{2k} \left(\binom{2k-1}{i-1} - \binom{2k-1}{i}\right) p^i q^{2k+1-i} + \sum_{i=k}^{2k} \left(\binom{2k-1}{i-1} - \binom{2k-1}{i}\right) p^{i+1} q^{2k-i} \\ &= \sum_{i=k}^{2k} \left(\binom{2k-1}{i-1} - \binom{2k-1}{i}\right) \left(p^i q^{2k+1-i} + p^{i+1} q^{2k-i}\right) - \left(\binom{2k-1}{k-1} - \binom{2k-1}{k}\right) p^k q^{k+1} \\ &= \sum_{i=k}^{2k} \left(\binom{2k-1}{i-1} - \binom{2k-1}{i}\right) p^i q^{2k-i} = \frac{A(2k,p,q)}{2k}, \end{aligned}$$

where we have used in the last step that $p+q=1$ and $\binom{2k-1}{k-1} = \binom{2k-1}{k}$. □

Lemma 26 is not an artifact of the approximation of the drift by function A but also holds for the exact drift-maximizing function B. This lemma will not be needed in the following, but we believe it to be interesting in its own right. The reader only interested in the proof of the main results of this work can skip this proof.

Lemma 27 (flipping even numbers of bits is sub-optimal, statement for exact drift B)**.** *For all $n, d, k \in \mathbb{N}$ satisfying $0 < d \leq \frac{n}{2}$ and $0 < 2k+1 \leq n$, it holds that $B(n,d,2k) < B(n,d,2k+1)$. Moreover, $\frac{B(n,d,2k)}{2k} = \frac{B(n,d,2k+1)}{2k+1}$ holds.*

Proof. Using again the shorthand $(r)_i := r(r-1)\cdots(r-i+1)$ for all positive integers r and $i \leq r$, we get

$$\begin{aligned} B(n,d,2k+1) &= \frac{1}{\binom{n}{2k+1}} \sum_{i=k+1}^{2k+1} \binom{d}{i}\binom{n-d}{2k-i+1}(2i-2k-1) \\ &= \frac{1}{\binom{n}{2k+1}} \sum_{i=0}^{k} \binom{d}{i+k+1}\binom{n-d}{k-i}(2i+1) \end{aligned}$$

$$= \frac{(n-2k-1)!}{n!}\sum_{i=0}^{k}\binom{2k+1}{k-i}(d)_{i+k+1}(n-d)_{k-i}(2i+1).$$

Similarly, we obtain

$$\begin{aligned}
B(n,d,2k) &= \frac{1}{\binom{n}{2k}}\sum_{i=k+1}^{2k}\binom{d}{i}\binom{n-d}{2k-i}(2i-2k)\\
&= \frac{1}{\binom{n}{2k}}\sum_{i=0}^{k-1}\binom{d}{i+k+1}\binom{n-d}{k-i-1}(2i+2)\\
&= \frac{(n-2k)!}{n!}\sum_{i=0}^{k-1}\binom{2k}{k-i-1}(d)_{i+k+1}(n-d)_{k-i-1}(2i+2).
\end{aligned}$$

Therefore, the ratio of $B(n,d,2k)$ and $B(n,d,2k+1)$ is

$$\begin{aligned}
\frac{B(n,d,2k)}{B(n,d,2k+1)} &= \frac{(n-2k)\sum_{i=0}^{k-1}\binom{2k}{k-i-1}(d)_{i+k+1}(n-d)_{k-i-1}(2i+2)}{\sum_{i=0}^{k}\binom{2k+1}{k-i}(d)_{i+k+1}(n-d)_{k-i}(2i+1)}\\
&= \frac{(n-2k)\sum_{i=0}^{k-1}\binom{2k}{k-i-1}(d-k-1)_i(n-d)_{k-i-1}(2i+2)}{\sum_{i=0}^{k}\binom{2k+1}{k-i}(d-k-1)_i(n-d)_{k-i}(2i+1)}.
\end{aligned}$$

Replacing $(d-k)$ with u and $(n-d)$ with v gives

$$\frac{B(n,d,2k)}{B(n,d,2k+1)} = \frac{(u+v-k)\sum_{i=0}^{k-1}\binom{2k}{k-i-1}(u-1)_i(v)_{k-i-1}(2i+2)}{\sum_{i=0}^{k}\binom{2k+1}{k-i}(u-1)_i(v)_{k-i}(2i+1)}.$$

Using the shorthand λ_i^j for the coefficient of r^j in the polynomial $(r)_i$, we now take a close look at the denominator, which is a polynomial in u and v. It is not difficult to see that for all $a,b\in\mathbb{N}\cup\{0\}$, the coefficient of the term u^av^b in the denominator equals

$$\psi(a,b) = \sum_{i=0}^{k}\binom{2k+1}{k-i}\lambda_{i+1}^{a+1}\lambda_{k-i}^{b}(2i+1),$$

while the coefficient of term u^av^b in numerator equals

$$\phi(a,b) = \sum_{i=0}^{k-1}\binom{2k}{k-i-1}\left(\lambda_{i+1}^{a}\lambda_{k-i-1}^{b}+\lambda_{i+1}^{a+1}\lambda_{k-i-1}^{b-1}-k\lambda_{i+1}^{a+1}\lambda_{k-i-1}^{b}\right)(2i+2).$$

Since $(r)_{i+1}=(r-i)(r)_i$, it is easily verified that

$$\lambda_i^j - i\lambda_i^{j+1} = \lambda_{i+1}^{j+1}. \tag{24}$$

We use (24) to simplify $\phi(a,b)$ in the following way.

$$\begin{aligned}
\phi(a,b) &= \sum_{i=0}^{k-1}\binom{2k}{k-i-1}\left(\lambda_{i+2}^{a+1}\lambda_{k-i-1}^{b}+\lambda_{i+1}^{a+1}\lambda_{k-i}^{b}\right)(2i+2)\\
&= \sum_{i=0}^{k}\left(\binom{2k}{k-i-1}(2i+2)+\binom{2k}{k-i}(2i)\right)\lambda_{i+1}^{a+1}\lambda_{k-i}^{b}
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{k} \left((2i+2) + \frac{k+i+1}{k-i}(2i) \right) \binom{2k}{k-i-1} \lambda_{i+1}^{a+1} \lambda_{k-i}^{b} \\
&= \sum_{i=0}^{k} \frac{2k(2i+1)}{k-i} \binom{2k}{k-i-1} \lambda_{i+1}^{a+1} \lambda_{k-i}^{b} \\
&= \frac{2k}{2k+1} \sum_{i=0}^{k} \binom{2k+1}{k-i} \lambda_{i+1}^{a+1} \lambda_{k-i}^{b} (2i+1) \\
&= \frac{2k}{2k+1} \psi(a,b).
\end{aligned}
$$

The above holds for all $0 \leq a, b \leq k$, showing that indeed

$$
\frac{B(n,d,2k)}{B(n,d,2k+1)} = \frac{2k}{2k+1}.
$$

□

4.3.2 Monotonicity of $\tilde{R}_{\mathrm{opt},\varepsilon}$

We now argue that, for all $\varepsilon > 0$, the function $\tilde{R}_{\mathrm{opt},\varepsilon}$ is monotone. It seems quite intuitive that the optimal number of bit flips should decrease with decreasing distance to the optimum, and this has been previously observed empirically, e.g., in [Bäc92, FCSS08, FCSS09]. However, formally proving the desired monotonic relationship requires substantial technical work. We note that, as a side result, Lemma 32 shows that for search points having a distance of less than $n/3$ to the optimum (or its complement), the maximal approximated fitness gain is obtained by 1-bit flips.

Lemma 28 (and definition of cut-off points)**.** *For any two integers $0 \leq k_1 < k_2$, the functions $p \mapsto A(2k_1+1, p, 1-p)$ and $p \mapsto A(2k_2+1, p, 1-p)$ intersect exactly once in the interval $(0, 1/2]$. Denoting this intersection p_0 and letting $A_0 := A(2k_1+1, p_0, 1-p_0)$, we call (p_0, A_0) the* **cut-off point** *of $A(2k_1+1, p, 1-p)$ and $A(2k_2+1, p, 1-p)$.*

We have $A(2k_1+1, p, 1-p) > A(2k_2+1, p, 1-p)$ if and only if $0 < p < p_0$.

The graph in Figure 1 illustrates the functions $p \mapsto A(k, p, 1-p)$ for $k = 1, 3, 5, 7$. The precise cut-off points will be computed numerically in Section 5.2.

In order to prove Lemma 28, we first show the following combinatorial lemma.

Lemma 29. *For all $k, r \in \mathbb{N} \cup \{0\}$ and $0 \leq q \leq 1$ it holds that*

$$
\sum_{i=0}^{k} \binom{k+r+1}{i} (1-q)^{k-i} q^i = \sum_{i=0}^{k} \binom{i+r}{r} q^i. \tag{25}
$$

Proof. We prove the equation by induction. It is obvious that equation (25) holds for all $r \geq 0$ and $k = 0$. Assume that it holds for some pair of integers $(k, r+1)$, then the following computation shows that it also holds for $(k+1, r)$.

$$
\begin{aligned}
&\sum_{i=0}^{k+1} \binom{k+1+r+1}{i} (1-q)^{k-i+1} q^i \\
&= \binom{k+r+2}{k+1} q^{k+1} + (1-q) \sum_{i=0}^{k} \binom{k+(r+1)+1}{i} (1-q)^{k-i} q^i
\end{aligned}
$$

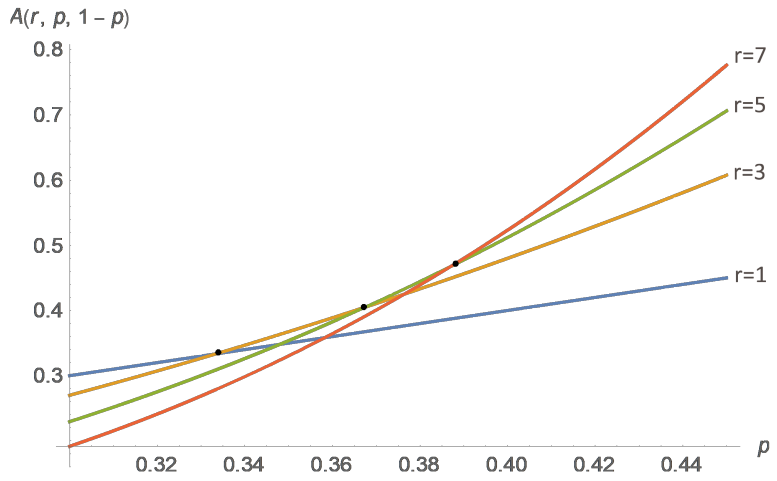

Figure 1: The approximated drift $A(k, p, 1-p)$ for $k = 1, 3, 5, 7$. By Lemma 32 the function $r \mapsto A(r, p, 1-p)$ is maximized for $r = 1$ whenever $p < 1/3$.

$$
\begin{aligned}
&= \binom{k+r+2}{r+1} q^{k+1} + (1-q) \sum_{i=0}^{k} \binom{i+r+1}{r+1} q^i \\
&= \sum_{i=0}^{k+1} \binom{i+r}{r} q^i.
\end{aligned}
$$

For arbitrary combinations of k and r, we thus get the desired correctness of (25) for the pair (k, r) inductively from that of the pair $(0, r+k)$. □

We use Lemma 29 to compute the second derivative of $A(r, p, q)$ for q in Lemma 30 and then obtain the second derivative of $A(r, p, q)$ for p in Lemma 31. We first notice that $A(1, p, 1-p) = p$ and $\mathrm{d}A(1, p, 1-p)/\mathrm{d}p = 1$. Thus we only look at the second derivative for $r > 1$.

Lemma 30. *For all $k \in \mathbb{N}$ and all $0 < p \leq 1/2$, it holds that*

$$\frac{\mathrm{d}^2 A(2k+1, p, 1-p)}{(\mathrm{d}(1-p))^2} = c_k p^{k-1} (1-p)^{k-1}, \tag{26}$$

where c_k is a constant related to k via

$$c_k := 2(2k-1)(2k+1)\binom{2k-2}{k-1} = \frac{4k+2}{\beta(k,k)},$$

and $\beta(x, y) := \int_0^1 t^{x-1}(1-t)^{y-1} dt = \frac{\Gamma(x)\Gamma(y)}{\Gamma(x+y)}$ is the well-known beta function.

Proof. Set $q := 1-p$. We expand $A(2k+1, p, q)$ according to Equation (23) and use Lemma 29 to obtain the following

$$
\begin{aligned}
A(2k+1, p, q) &= (2k+1) \sum_{i=k}^{2k} \left(\binom{2k}{i} - \binom{2k}{i+1} \right) p^{i+1} q^{2k-i} \\
&= (2k+1) \sum_{i=0}^{k} \left(\binom{2k}{k+i} - \binom{2k}{k+i+1} \right) p^{k+i+1} q^{k-i}
\end{aligned}
$$

$$
\begin{aligned}
&= (2k+1)p^{k+1}\cdot\sum_{i=0}^{k}\left(\binom{2k}{i}-\binom{2k}{i-1}\right)p^{k-i}q^i\\
&= (2k+1)p^{k+1}\left[\sum_{i=0}^{k}\binom{k+(k-1)+1}{i}p^{k-i}q^i+q\sum_{i=0}^{k-1}\binom{(k-1)+k+1}{i}p^{k-1-i}q^i\right]\\
&= (2k+1)p^{k+1}\left[\sum_{i=0}^{k}\binom{i+(k-1)}{i}q^i+q\sum_{i=0}^{k-1}\binom{i+k}{i}q^i\right]\\
&= (2k+1)p^{k+1}\cdot\sum_{i=0}^{k}\left(\binom{k+i-1}{i}-\binom{k+i-1}{i-1}\right)q^i.
\end{aligned}
$$

We extract the term p^{k+1} in the above equation and define the polynomial $f_k(q) := \sum_{i=0}^{\infty} a_k^i q^i$ with coefficient $a_k^i = 0$ when $i > k$ and

$$
a_k^i := \binom{k+i-1}{i}-\binom{k+i-1}{i-1} = \binom{k+i-1}{i}\frac{k-i}{k} \text{ when } i \le k. \tag{27}
$$

Then $A(2k+1,p,q) = (2k+1)p^{k+1}f_k(q)$. We use the general Leibniz rule for the second derivative (informally, this rule states that $(fg)'' = f''g + 2f'g' + fg''$) and obtain

$$
\frac{\mathrm{d}^2A(2k+1,p,q)}{\mathrm{d}^2q} = (2k+1)p^{k-1}\cdot\left(p^2f_k''(q)-2(k+1)pf_k'(q)+(k+1)kf_k(q)\right).
$$

It remains to prove that

$$
p^2f_k''(q)-2(k+1)pf_k'(q)+(k+1)kf_k(q) = 2(2k-1)\binom{2k-2}{k-1}q^{k-1}. \tag{28}
$$

We look at the coefficient of q^i in the left part of equation (28) and we denote it by c_k^i. By expanding $f_k(q)$ into a polynomial and replacing p by $1-q$, we see that c_k^i equals the coefficient of q^i in the following expression

$$
(1-q)^2(a_k^iq^i+a_k^{i+1}q^{i+1}+a_k^{i+2}q^{i+2})''-2(k+1)(1-q)(a_k^iq^i+a_k^{i+1}q^{i+1})'+(k+1)k(a_k^iq^i).
$$

This shows that c_k^i is equal to

$$
a_k^i\left((k+1)k+2(k+1)i+i(i-1)\right)+a_k^{i+1}\left(-2(k+1)(i+1)-2(i+1)i\right)+a_k^{i+2}\left((i+2)(i+1)\right).
$$

According to (27) the coefficient a_k^i satisfies

$$
a_k^{i+1} = a_k^i + a_{k-1}^{i+1}, \text{ and} \tag{29}
$$

$$
a_{k-1}^{i+1} = a_k^i\cdot\frac{k}{i+1}\cdot\frac{k-i-2}{k-i}, \text{ for } k > i. \tag{30}
$$

We use Equation (29) to rewrite the expression of c_k^i for $i \le k-2$, and then use Equation (30) to simplify the equation in the following way

$$
\begin{aligned}
c_k^i &= a_k^i\left(k(k-1)\right)+a_{k-1}^{i+1}\left(-2(k-1)(i+1)\right)+a_{k-2}^{i+2}\left((i+2)(i+1)\right)\\
&= a_k^ik(k-1)-2a_k^ik(k-1)\frac{k-i-2}{k-i}+a_k^ik(k-1)\frac{k-i-4}{k-i}\\
&= 0.
\end{aligned}
$$

Noticing that $a_k^k = 0$ shows that $f_k(q)$ has a degree of $k-1$. This implies that the term q^{k-1} has the highest degree in (28). Its coefficient is

$$\begin{aligned} c_k^{k-1} &= a_k^{k-1}\left((k+1)k + 2(k+1)(k-1) + (k-1)(k-2)\right) \\ &= 2(2k-1)\binom{2k-2}{k-1}. \end{aligned}$$

This proves the claimed equality in (26). □

Lemma 31. *For all $k \in \mathbb{N}$ and all $0 < p \leq 1/2$, it holds that*

$$\begin{aligned} \frac{\mathrm{d}^2 A(2k+1,p,1-p)}{\mathrm{d}p^2} &= c_k p^{k-1}(1-p)^{k-1}, \\ \frac{\mathrm{d}A(2k+1,p,1-p)}{\mathrm{d}p} &= c_k \int_0^p x^{k-1}(1-x)^{k-1}\mathrm{d}x \text{ and} \\ A(2k+1,p,1-p) &= c_k \int_0^p \int_0^y x^{k-1}(1-x)^{k-1}\mathrm{d}x\mathrm{d}y. \end{aligned}$$

Furthermore for all $k \in \mathbb{N}_0$ we can write

$$A(2k+1,p,1-p) = \int_0^p \frac{\mathrm{d}A(2k+1,x,1-x)}{\mathrm{d}p}\mathrm{d}x.$$

Proof. The first equality can be easily obtained from the equality in (26). Consequently,

$$\begin{aligned} \frac{\mathrm{d}A(2k+1,p,q)}{\mathrm{d}p} &= c_k \int_0^p x^{k-1}(1-x)^{k-1}\mathrm{d}x + C_1 \text{ with } C_1 \in \mathbb{R}, \\ A(2k+1,p,q) &= c_k \int_0^p \int_0^y x^{k-1}(1-x)^{k-1}\mathrm{d}x\mathrm{d}y + C_1 p + C_2 \text{ with } C_2 \in \mathbb{R}. \end{aligned}$$

Using the fact that $\lim_{p\to 0} A(2k+1,p,q) = o(p)$ for $k \geq 1$, we obtain $C_1 = C_2 = 0$ as claimed.

For the last statement, we only need to consider the case $k = 0$. Recalling that $A(1,p,1-p) = p$ and $\mathrm{d}A(1,p,1-p)/\mathrm{d}p = 1$ shows that the equality also applies to this case. □

We next prove Lemma 28.

Proof of Lemma 28. Using the notation from Lemma 30, we first notice that for all $k > 0$ we have

$$\frac{c_{k+1}}{c_k} = \frac{2(2k+1)(2k+3)\binom{2k}{k}}{2(2k-1)(2k+1)\binom{2k-2}{k-1}} = \frac{4k+6}{k} > 4. \tag{31}$$

Let $0 < k_1 < k_2$. By the above, we have $4 < (4+6/k_2)^{k_2-k_1} < c_{k_2}/c_{k_1} < (4+6/k_1)^{k_2-k_1}$. Notice that $c_{k_1}(pq)^{k_1-1} - c_{k_2}(pq)^{k_2-1} = (pq)^{k_1-1}(c_{k_1} - c_{k_2}(pq)^{k_2-k_1})$. We now use the fact that $\lim_{p\to 0} pq = 0$ and $\lim_{p\to 1/2} pq = 1/4$ to obtain that for all $k_2 > k_1 > 0$,

$$\lim_{p\to 0}(c_{k_1} - c_{k_2}(pq)^{k_2-k_1}) > 0 \text{ while } \lim_{p\to 1/2}\left(c_{k_1} - c_{k_2}(pq)^{k_2-k_1}\right) < 0.$$

This shows that the function $p \mapsto c_{k_1}(pq)^{k_1-1}$ intersects with $p \mapsto c_{k_2}(pq)^{k_2-1}$ in at most one point $p_I \in (0,1/2)$. Moreover, we have that $c_{k_1}(pq)^{k_1-1} \geq c_{k_2}(pq)^{k_2-1}$ if and only if $p \in [0,p_I]$. Therefore, when $p > 0$, the function $p \mapsto \int_0^p c_{k_1}(x-x^2)^{k_1-1}\mathrm{d}x$ intersects with the function $p \mapsto \int_0^p c_{k_2}(x-x^2)^{k_2-1}\mathrm{d}x$ at most once for $p \in (0,1/2)$. We now prove that the intersection exists.

Notice that for all $k > 1$ we have

$$c_k \int_0^{0.5} (x - x^2)^{k-1} \mathrm{d}x = \frac{c_k}{2} \int_0^1 (x - x^2)^{k-1} \mathrm{d}x = \frac{c_k}{2} \beta(k, k) = 2k + 1,$$

and thus

$$\lim_{p \to 1/2} \left(\int_0^p c_{k_1} (x - x^2)^{k_1 - 1} \mathrm{d}x - \int_0^p c_{k_2} (x - x^2)^{k_2 - 1} \mathrm{d}x \right) = 2(k_1 - k_2) < 0,$$

while

$$\int_0^p c_{k_1} (x - x^2)^{k_1 - 1} \mathrm{d}x > \int_0^p c_{k_2} (x - x^2)^{k_2 - 1} \mathrm{d}x \text{ for all } p \in (0, p_I).$$

There exists a intersection point $p_{II} \in (0, 1/2)$ such that

$$\int_0^p c_{k_1} (x - x^2)^{k_1 - 1} \mathrm{d}x \geq \int_0^p c_{k_2} (x - x^2)^{k_2 - 1} \mathrm{d}x \text{ if and only if } p \in [0, p_{II}].$$

This shows that for all $k_2 > k_1 > 0$ there exists a point $p_{II} \in (0, 1/2)$ such that

$$\frac{\mathrm{d}A(2k_1 + 1, p, q)}{\mathrm{d}p} \geq \frac{\mathrm{d}A(2k_2 + 1, p, q)}{\mathrm{d}p} \text{ if and only if } p \in [0, p_{II}]. \tag{32}$$

To extend the conclusion to $k_1 = 0$, let $k_2 > k_1 = 0$. We have $\lim_{p \to 1/2} \int_0^p c_{k_2} (x - x^2)^{k_2 - 1} dx = 2k_2 + 1$ and $\lim_{p \to 0} \int_0^p c_{k_2} (x - x^2)^{k_2 - 1} dx = 0$, while $\mathrm{d}A(1, p, q)/\mathrm{d}p = 1$. Therefore the intersection point p_{II} still exists and there is a unique such point.

As a result we see that the function $\int_0^p \mathrm{d}A(2k_1 + 1, x, 1 - x)$ intersects with the function $\int_0^p \mathrm{d}A(2k_2 + 1, x, 1 - x)$ at most once for $p \in (0, 1/2)$ and

$$\int_0^p \mathrm{d}A(2k_1 + 1, x, q) > \int_0^p \mathrm{d}A(2k_2 + 1, x, q) \text{ for all } p \in (0, p_{II}) \text{ while}$$
$$\lim_{p \to 1/2} \left(A(2k_1 + 1, p, q) - A(2k_2 + 1, p, q) \right) < 0.$$

This shows that $A(2k_1 + 1, p, q)$ intersects with $A(2k_2 + 1, p, q)$ exactly once at some value $p_{III} < 1/2$.

□

We are now ready to prove the monotonicity of $\tilde{R}_{\mathrm{opt},\varepsilon}$.

Proof of the first part of Theorem 25. Let $\varepsilon > 0$, let $p_0 \in (0, 1)$, and set $q_0 := 1 - p_0$. By Lemma 26 it holds that $A(2k, p_0, q_0) < A(2k + 1, p_0, q_0)$. This shows that $\tilde{R}_{\mathrm{opt},\varepsilon}(p_0)$ is odd. Let $k \in \mathbb{N} \cup \{0\}$ such that $\tilde{R}_{\mathrm{opt},\varepsilon}(p_0) = 2k + 1$. By definition of $\tilde{R}_{\mathrm{opt},\varepsilon}$ (cf. equation (22)), k is the smallest integer obtaining a drift of $A(2k + 1, p_0, q_0)$. For all integers $k' < k$ we thus obtain

$$A(2k + 1, p_0, q_0) > A(2k' + 1, p_0, q_0). \tag{33}$$

By Lemma 28 we also get that for all $p > p_0$ it holds that

$$A(2k + 1, p, q) > A(2k' + 1, p, q). \tag{34}$$

Therefore $\tilde{R}_{\mathrm{opt},\varepsilon}(p) \geq 2k + 1$ for all $p > p_0$. Since the statement holds for all $p_0 \in (0, 1/2 - \varepsilon]$ we obtain the monotonicity of $\tilde{R}_{\mathrm{opt},\varepsilon}$. □

4.3.3 $\tilde{R}_{\mathrm{opt},\varepsilon}(p) = 1$ when $0 < p < 1/3$ and $R_{\mathrm{opt}}(d,n) = 1$ when $0 < d = o(n)$

We first show that flipping one bit is optimal for the approximated drift when the distance to the optimal solution is less than $n/3$.

Lemma 32. *For all $\varepsilon > 0$ and all $0 < p < 1/3$ it holds that $\tilde{R}_{\mathrm{opt},\varepsilon}(p) = 1$.*

Proof. Let $\varepsilon > 0$. Due to the monotonicity of $\tilde{R}_{\mathrm{opt},\varepsilon}$, it suffices to show that $\tilde{R}_{\mathrm{opt},\varepsilon}(1/3) = 1$. By Lemma 26 we only need to consider odd values of r. According to Lemma 31 if the second derivative $c_k x^{k-1}(1-x)^{k-1} > c_{k+1}x^k(1-x)^k$ for all $x \in (0,1/3)$ then $A(2k-1,1/3,2/3) > A(2k+1,1/3,2/3)$ and thus $\tilde{R}_{\mathrm{opt},\varepsilon} \neq 2k+1$. We notice that

$$\frac{c_{k+1}\ x^k(1-x)^k}{c_k\ x^{k-1}(1-x)^{k-1}} = \frac{4k+6}{k}x(1-x) \leq \frac{4k+6}{k}\cdot\frac{2}{9} \text{ for all } 0 < x < \frac{1}{3}.$$

For all $k > 12$ it holds that $(4k+6)/k < 9/2$, which implies that $c_k x^{k-1}(1-x)^{k-1} > c_{k+1}x^k(1-x)^k$. We therefore obtain that $\tilde{R}_{\mathrm{opt},\varepsilon}(1/3) \leq 25$. For the remaining values, i.e., for $r = 1,3,\dots,25$, we can compute $A(r,1/3,2/3)$ numerically. This numerical evaluation shows that the maximum value $1/3$ is obtained (only) by $r = 1$ and $r = 3$. This proves $\tilde{R}_{\mathrm{opt},\varepsilon}(1/3) = 1$. □

We show in Lemma 33 that flipping one bit is also optimal for the exact fitness drift when the distance is a lower-order term of n.

Lemma 33. *For all $0 < d = o(n)$ it holds that $R_{\mathrm{opt}}(d,n) = 1$.*

Proof. Since $d < n/4$, Lemma 20 yields with $\varepsilon := 1/4$ that $R_{\mathrm{opt}}(d,n) < 4^4\log(4)$. Referring to Lemma 23, we obtain for all $3 \leq r \leq 4^4\log(4)$ that $B(n,d,r) < A(r,d/n,1) = \Theta((d/n)^{(r+1)/2}) = o(d/n)$. Since R_{opt} attains only odd values, we obtain $R_{\mathrm{opt}}(d,n) = 1$ for all $0 < d = o(n)$. □

4.4 Runtime Loss From Using the Approximated Drift

We show in this section that the expected runtimes of the exact and the approximate drift maximizer do not differ substantially. More precisely, we show that also the approximate drift maximizer also obtains an expected runtime on OneMax that is very close to that of an optimal unary unbiased black-box algorithm, cf. Corollary 35. To make things precise, we denote for every $\varepsilon > 0$ by $\tilde{A}^*_\varepsilon$ the algorithm which we obtain from Algorithm 3 by replacing the mutation rate $R(\mathrm{OM}(x))$ by $\tilde{R}_{\mathrm{opt},\varepsilon}(1-\mathrm{OM}(x)/n)$.

To state the main result, for all $\varepsilon > 0$, for all $n \in \mathbb{N}$, and all $0 < p \leq 1/2 - \varepsilon$ we abbreviate

$$A_{\max,\varepsilon}(p) := A(\tilde{R}_{\mathrm{opt},\varepsilon}(p), p, 1-p) \text{ and } B_{\max}(p,n) := B(n, \lfloor pn \rfloor, R_{\mathrm{opt}}(\lfloor pn \rfloor, n)). \tag{35}$$

We notice from Lemma 20 and Lemma 24 that $\tilde{R}_{\mathrm{opt},\varepsilon}(1/2-\varepsilon) = \Theta(1)$ and $R_{\mathrm{opt}}(\lfloor(1/2-\varepsilon)n\rfloor, n) = \Theta(1)$. Considering the drift of single bit flip, we see that $A_{\max,\varepsilon}(1/2-\varepsilon) \geq 1/2-\varepsilon$. According to the definition of $\tilde{h}$ in equation (6), we have $B_{\max}(d/n,n) = \tilde{h}(d) \geq d/n$ for all $0 < d \leq (1/2-\varepsilon)n$.

Theorem 34. *For all constant $0 < \varepsilon < 1/2$ the expected runtime of algorithm $\tilde{A}^*_\varepsilon$ on OneMax satisfies*

$$E\left(T_{\tilde{A}^*_\varepsilon}\right) \leq \sum_{x=1}^{(1/2-\varepsilon)n} \frac{1}{h(x)} + \frac{\varepsilon n}{A_{\max,\varepsilon}(1/2-\varepsilon)} + o(n).$$

Moreover,

$$E\left(T_{\tilde{A}^*_\varepsilon}\right) \leq \sum_{x=1}^{(1/2-\varepsilon)n} \frac{1}{A_{\max,\varepsilon}(x/n)} + \frac{\varepsilon n}{A_{\max,\varepsilon}(1/2-\varepsilon)} + o(n).$$

Proof. Let constant ε be a constant with $0 < \varepsilon < 1/2$. It is easily seen from Theorem 7 and from the definition of $\tilde{R}_{\text{opt},\varepsilon}$ in (22) that

$$E\left(T_{\tilde{A}^*_\varepsilon} \mid x(0)\right) \leq \sum_{x=1}^{n-\text{OM}(x(0))} \frac{1}{B(n,x,\tilde{R}_{\text{opt},\varepsilon}(x/n))} \leq \sum_{x=1}^{n/2} \frac{1}{B(n,x,\tilde{R}_{\text{opt},\varepsilon}(x/n))} + 1. \tag{36}$$

To ease representation, let $r_A(x) := \tilde{R}_{\text{opt},\varepsilon}(x/n)$ and $r_B(x) := R_{\text{opt}}(x,n)$ for all $0 < x \leq (1/2-\varepsilon)n$. According to Theorem 22 we have

$$\left|A\left(r_A(x), \frac{x}{n}, 1-\frac{x}{n}\right) - B\left(n,x,r_A(x)\right)\right| = O(1/x) \text{ and } \left|A\left(r_B(x), \frac{x}{n}, 1-\frac{x}{n}\right) - B(n,x,r_B(x))\right| = O(1/x).$$

Since $A(r_B(x), x/n, 1-x/n) \leq A(r_A(x), x/n, (n-x)/n) = A_{\max,\varepsilon}(x/n)$ and $B(n,x,r_A(x)) \leq B(n,x,r_B(x)) = B_{\max}(x/n,n) = \tilde{h}(x)$, we obtain from Theorem 22 that, for all $0 < x < (1/2-\varepsilon)n$,

$$\begin{aligned} \left|B(n,x,r_A(x)) - \tilde{h}(x)\right| &= O(1/x) \text{ and} \\ |A_{\max,\varepsilon}(x/n) - B(n,x,r_A(x))| &= O(1/x), \end{aligned}$$

where we use the fact that $r_A(x) = \Theta(1)$ according to Lemma 24. Referring to Lemma 32 and Lemma 33, we have $r_B(x) = r_A(x) = 1$ when $0 < x = o(n)$. Therefore,

$$A_{\max,\varepsilon}(x/n) = B(n,x,r_A(x)) = \tilde{h}(x) \text{ for all } 0 < x = o(n).$$

Notice that $h(x) \geq B(n,x,1) = x/n$ for all $x > 0$. Using the fact that $0 < h(x) - \tilde{h}(x) \leq n\exp(-\Omega(n^{0.2}))$ for $0 < x < n/2 - n^{0.6}$ in Lemma 16, we obtain

$$\left|\frac{1}{B(n,x,r_A(x))} - \frac{1}{h(x)}\right| = \frac{|h(x) - B(n,x,r_A(x))|}{B(n,x,r_A(x))h(x)} \leq \begin{cases} o(1/n) \text{ for } 0 < x \leq n^{0.99}, \\ O((1/x)/(x/n)^2) \text{ for } n^{0.99} < x \leq (1/2-\varepsilon)n. \end{cases}.$$

Referring to Lemma 18 and the definition of $B(n,x,r)$ we see that, for all fixed n and r, the fitness drift $B(n,x,r)$ monotonically increases with respect to x. Therefore, for all $x > (1/2-\varepsilon)n$, we have $r_A(x) = r_A((1/2-\varepsilon)n)$ and $B(n,x,r_A(x)) \geq B(n,(1/2-\varepsilon)n,r_A((1/2-\varepsilon)n)) \geq A_{\max,\varepsilon}(1/2-\varepsilon) - O(1/n) \geq 1/2 - \varepsilon - O(1/n) = \Omega(1)$, thus

$$E(T_{\tilde{A}^*_\varepsilon}) \leq \sum_{x=1}^{(1/2-\varepsilon)n} \frac{1}{h(x)} + O(1) + O(n^{0.03}) + \frac{\varepsilon n}{A_{\max,\varepsilon}(1/2-\varepsilon)} + o(n).$$

This proves the first statement.

The second statement can be shown by using similar methods to bound the absolute difference $|1/B(n,x,r_A(x)) - 1/A_{\max,\varepsilon}(x/n)|$ for $0 < x \leq n/2$. □

Corollary 35. *For all constant ε with $0 < \varepsilon < 1/2$, the difference between the expected runtime of $\tilde{A}^*_\varepsilon$ on* OneMax *and that of an optimal unary unbiased black-box algorithm is $O(\varepsilon n)$. Furthermore, the absolute difference between the expected runtimes of A^* and $\tilde{A}^*_\varepsilon$ is also $O(\varepsilon n)$.*

Proof. Let constant $0 < \varepsilon < 1/2$ and let A be an arbitrary unary unbiased black-box algorithm. By Theorems 14 and 34 it holds that

$$E\left(T_A\right) \geq \sum_{x=1}^{n/2-n^{0.6}} \frac{1}{h(x)} - \Theta(n^{2/3}\ln^9(n))$$

$$\geq \sum_{x=1}^{(1/2-\varepsilon)n} \frac{1}{h(x)} - o(n)$$
$$\geq E\left(T_{\tilde{A}^*_\varepsilon}\right) - O(\varepsilon n).$$

The second statement is a direct consequence of the first and Theorem 11. □

5 Runtime Analysis for the Approximate Drift-Maximizer $\tilde{A}^*_\varepsilon$

We compute in this section the expected time needed by Algorithm $\tilde{A}^*_\varepsilon$ to optimize OneMax. We fix $0 < \varepsilon < 1/2$. As proven in Lemma 32 algorithm $\tilde{A}^*_\varepsilon$ flips $\tilde{R}_{\mathrm{opt},\varepsilon}(p) = 1$ bit whenever $0 < p < 1/3$. In this regime $\tilde{A}^*_\varepsilon$ is thus equal to RLS. It is well known (and easy to prove by a simple fitness-level argument) that the expected time needed by RLS starting in a search point of OneMax value $2n/3$ to reach the all-ones string equals $n\sum_{i=1}^{n/3} 1/i = nH_{n/3} = n(\ln(n/3) + \gamma) + 3/2 + O(1/n)$, where $\gamma \approx 0.57721\ldots$ denotes again the EulerMascheroni constant. It therefore remains to compute the time needed by $\tilde{A}^*_\varepsilon$ to reach for the first time a search point having fitness at least $2n/3$.

Formally, we also need to show that the first search point having fitness at least $2n/3$ does not have a fitness value that is much larger than this. Since we flip a constant number of bits only, we get this statement for free. Note also that it is shown below that in the interval before reaching this fitness level the algorithm flips only 3 bits. Apart from this situation around fitness layer $2n/3$ we do not have to take care of jumping several fitness layers by hand, but this is taken into account already in the drift theorems from which we derive our runtime estimates.

5.1 Drift Analysis

As we did in Section 3, we employ the variable drift theorems, Theorems 7 and 10, to compute upper and lower bounds for the expected runtime of algorithm $\tilde{A}^*_\varepsilon$. We will provide a numerical evaluation of these expressions in Section 5.2.

Lower bound. We first compute a lower bound for the expected runtime $E(T_A)$ of any unary unbiased black-box algorithm A on OneMax. According to Theorem 14 and using a similar method to estimate $|1/A_{\max,\varepsilon}(x/n) - 1/h(x)| = o(1)$ as in Theorem 34 for $0 < x \leq (1/2-\varepsilon)n$, we obtain that

$$\begin{aligned} E(T_A) &\geq \sum_{x=1}^{n/2-n^{0.6}} \frac{1}{h(x)} - \Theta(n^{2/3}\ln^9(n)) \geq \sum_{x=1}^{(1/2-\varepsilon)n} \frac{1}{A_{\max}(x/n)} - o(n) \\ &= nH_{n/3} + \sum_{x=\lfloor n/3\rfloor}^{(1/2-\varepsilon)n} \frac{1}{A_{\max}(x/n)} - o(n). \end{aligned}$$

Let $k \in \mathbb{N}$ and let $1/2-\varepsilon =: p_0 > p_1 > \ldots > p_k > 1/3$. Using the fact that $A_{\max,\varepsilon}$ is increasing, we bound $E(T_A)$ by

$$E(T_A) \geq n\left(\ln\left(\frac{n}{3}\right) + \gamma + \sum_{i=1}^{k} \frac{p_{i-1}-p_i}{A_{\max,\varepsilon}(p_{i-1})} + \int_{1/3}^{p_k} \frac{\mathrm{d}p}{A_{\max,\varepsilon}(p)}\right) - o(n). \tag{37}$$

Upper bound. Using the fact that $\tilde{R}_{\mathrm{opt},\varepsilon}(x/n) = 1$ for all $0 < x \leq n/3$ and referring to

r	L_r	R_r	$A_{\max,\varepsilon}(L_r)$	$A_{\max,\varepsilon}(R_r)$	$R_r - L_r$
3	0.333333333	0.367544468	0.333333	0.405267	0.034211135
5	0.367544468	0.386916541	0.405267	0.467174	0.019372073
7	0.386916541	0.399734261	0.467174	0.522084	0.012817721
9	0.399734261	0.409006003	0.522084	0.571870	0.009271741
11	0.409006003	0.416109983	0.571870	0.617718	0.007103980

Table 1: The optimal number of bit flips in interval $(L_r n, R_r n]$ is r.

Theorem 34, we obtain

$$E\left(T_{\tilde{A}^*_\varepsilon}\right) \leq nH_{n/3} + \sum_{x=\lfloor n/3 \rfloor}^{(1/2-\varepsilon)n} \frac{1}{A_{\max,\varepsilon}(x/n)} + \frac{\varepsilon n}{A_{\max,\varepsilon}(1/2-\varepsilon)} + o(n).$$

Using the same partition points as in the lower bound statement and the monotonicity of $A_{\max,\varepsilon}$, we have

$$E(T_{\tilde{A}^*_\varepsilon}) \leq n\left(\ln\left(\frac{n}{3}\right) + \gamma + \sum_{i=1}^{k} \frac{p_{i-1}-p_i}{A_{\max,\varepsilon}(p_i)} + \frac{1/2-p_0}{A_{\max,\varepsilon}(p_0)} + \int_{1/3}^{p_k} \frac{\mathrm{d}p}{A_{\max,\varepsilon}(p)}\right) + o(n). \tag{38}$$

5.2 Numerical Evaluation of the Expected Runtime

In this section we evaluate expressions (37) and (38) numerically to compute an estimate for the expected runtime of algorithm $\tilde{A}^*_\varepsilon$ on OneMax and for the unary unbiased black-box complexity.

Theorem 36. *For sufficiently small $\varepsilon > 0$ the expected runtime $E(T_{\tilde{A}^*_\varepsilon})$ of algorithm $\tilde{A}^*_\varepsilon$ on* OneMax *is $n\ln(n) - cn \pm o(n)$ for a constant c between 0.2539 and 0.2665. This bound is also the unary unbiased black-box complexity of* OneMax.

We can rewrite the expression in Theorem 36 to $n\left(\ln\left(n/3\right) + \gamma + c'\right) + o(n)$ for a constant c' between 0.2549 and 0.2675 to ease a comparison with the expected runtime of the previously best known unary unbiased algorithm, which is the one presented in [dPdLDD15]. This latter algorithm has an expected runtime equaling that of RLS up to an additive term of order $o(n)$. It is hence $n(\ln(n/2)+\gamma) \pm o(n)$. For sufficiently small $\varepsilon > 0$ Algorithm $\tilde{A}^*_\varepsilon$ is thus by an additive $(\ln(3) - \ln(2) - c')n \pm o(n)$ term faster, on average, than RLS or the algorithm presented and analyzed in [dPdLDD15]. That is, compared to RLS, algorithm $\tilde{A}^*_\varepsilon$ saves between $0.138n \pm o(n)$ and $0.151n \pm o(n)$ iterations on average.

To compute $E(T_{\tilde{A}^*_\varepsilon})$, we split the interval $(0, \frac{1}{2})$ into intervals $(L_{2i+1}, R_{2i+1}]$, $i = 0, 1, \ldots$, such that for each i and each $p \in (L_{2i+1}, R_{2i+1}]$ the number $\tilde{R}_{\mathrm{opt},\varepsilon}(p)$ of bits that need to be flipped in order to maximize the approximated expected fitness increase $A(\cdot, p, 1-p)$ is $2i+1$ (note that this is independent of ε, since ε just determines the cut-off point after which only use a bound for the drift-maximizing number of bit flips). Table 1 displays the first few intervals along with the corresponding drift values at the borders of the interval. We observe that the further we are away from the optimum (this corresponds to larger r by Theorem 25), the smaller the size of the interval.

The bound for the expected runtime of algorithm $\tilde{A}^*_\varepsilon$ reported in Theorem 36 is obtained by setting $p_0 = R_{4001}$, $p_k = R_9$ and using the following partition points

$$p_0 = R_{4001} > R_{3001} > R_{2001} > R_{1001} > R_{951} > R_{901} > R_{851} > \cdots > R_{200} > R_{151} > R_{101} > R_{35} > R_{34} > \cdots > R_{10} > R_9 = p_k.$$

The accuracy of our approximation can be increased by adding denser partition points, especially to the smaller side near p_k.

6 Fixed-Budget Analysis

In this section, we compare the algorithms developed in this work with the classic RLS heuristic (which was the essentially best previous unary unbiased algorithm for OM) in the *fixed-budget perspective*, that is, we compare the expected fitnesses obtained after a fixed budget B of iterations. This performance measure was introduced by Jansen and Zarges [JZ14] to reflect the fact that the most common use of search heuristics is not to compute an optimal solution, but only a solution of reasonable quality. We note that the time to reach a particular solution quality, called $T_{A,f}(a)$ in [DJWZ13] where this notion was first explicitly defined, would be an alternative way to phrase such results. We do not regard this performance measure here, but we would expect that, in a similar vein in as the following analysis, also in this measure our algorithm is superior to RLS by a (small) constant percentage.

Our main result in this section is that our drift maximizer with a fixed budget compute solutions having a roughly 13% smaller fitness distance to the optimum. This result contrasts the lower-order advantage in terms of the expected runtime, i.e., the average time needed to find an optimal solution.

The main challenge is proving the innocent statement that the time taken by our algorithm to find a solution x of fitness $\text{OM}(x) \geq 2n/3$ is strongly concentrated. Such difficulties occur often in fixed-budget analyses, see, e.g. [DJWZ13]. We prove the desired concentration via the following well-known martingale version of Azuma's inequality [Azu67] (as opposed to the simpler method of bounded differences, which appears not to be applicable here).

Theorem 37 (Method of Bounded Martingale Differences)**.** *Let $X_1, X_2, \dots, X_n$ be an arbitrary sequence of random variables and let f be a function satisfying the property that for each $i \in [n]$, there is a non-negative c_i such that $|E(f \mid X_0, X_1, \dots, X_{i-1}) - E(f \mid X_0, X_1, \dots, X_i)| \leq c_i$. Then*

$$\Pr\left(|f - E(f)| \geq \delta\right) \leq 2\exp\left(-\frac{\delta^2}{2\sum_{i=1}^n c_i^2}\right)$$

for all $\delta > 0$.

Consider a run of the algorithm $\tilde{A}^*_\varepsilon$ with small constant $0 < \varepsilon < 1/6$. Let $X_t := n - \max\{\text{OM}(x(i)) \mid i \in [0..t]\}$ be the current smallest fitness distance and let $r_{\max} := \tilde{R}_{\text{opt},\varepsilon}(1/2-\varepsilon)$ be the maximal mutation strength. Let $T_{1/3}$ be the first time at which the distance to the optimum is at most $n/3$, i.e., $T_{1/3}$ is the smallest t for which $X_t \leq n/3$. Let $N := 3r_{\max}n$ and define the function f by setting $f(X_0, X_1, \dots) := \min\{N, T_{1/3}\}$.

We notice that $\Pr(T_{1/3} > f) = \Pr(T_{1/3} > N) = \Pr(X_N > n/3)$. Referring to Lemma 22, we obtain for all $X_t > n/3$ that $E(X_t - X_{t+1} \mid X_t) = B(n, X_t, \tilde{R}_{\text{opt}}(X_t/n)) \geq A_{\max,\varepsilon}(X_t/n) - O(1/X_t) \geq 1/3 - o(1)$. Using the fact that $X_t - X_{t+1} \leq r_{\max}$, we obtain $\Pr(X_t > X_{t+1} \mid X_t > n/3) \geq 1/(3r_{\max}) - o(1)$. Define binary random variables Y_t by setting $Y_t := \mathbb{1}_{X_t > X_{t+1}}$, if $X_t > n/3$, and otherwise by having $Y_t = 1$ with probability $1/(3r_{\max}) - o(1)$ independently for all such Y_t. Note that, by definition, we have $\Pr[Y_t = 1] \geq 1/(3r_{\max}) - o(1)$ regardless of the outcomes of $Y_{t'}$, $t' < t$. Consequently, by well-known results, e.g., Lemma 3 in [Doe18a], the Y_t admit the same Chernoff bounds for the lower tail as independent binary random variables with success probability $1/(3r_{\max}) - o(1)$. We thus estimate $\Pr(X_N > n/3) \leq \Pr(Y_1 + Y_2 + \dots + Y_N < (2/3)n) = \exp(-\Omega(n))$. Consequently $E(T_{1/3} - f) < E(T)\Pr(T_{1/3} > f) = \exp(-\Omega(n))$.

Using the fact that $X_t - X_{t+1} \leq r_{\max}$, the additive drift theorem yields that the expected influence of one iteration on the remaining optimization time is at most $r_{\max}/(1/3 - o(1)) < 4r_{\max}$. Consequently, for $1 \leq i \leq N$, we have

$$|E(f \mid X_0, X_1, \cdots, X_{i-1}) - E(f \mid X_0, X_1, \cdots, X_i)| \leq 4r_{\max}.$$

Applying Theorem 37 and using the fact that $\Pr(T_{1/3} > f) = \exp(-\Omega(n))$ and $E(T_{1/3} - f) = \exp(-\Omega(n))$, we compute

$$\begin{aligned}
\Pr\left(|T_{1/3} - E(T_{1/3})| \geq n^{0.6}\right) &\leq \Pr\left(|f - E(T_{1/3})| \geq n^{0.6}\right) + \exp(-\Omega(n)) \\
&\leq \Pr\left(|f - E(f) - E(T_{1/3} - f)| \geq n^{0.6}\right) + \exp(-\Omega(n)) \\
&\leq \Pr\left(|f - E(f)| \geq n^{0.6} - E(T_{1/3} - f)\right) + \exp(-\Omega(n)) \\
&\leq \Pr\left(|f - E(f)| \geq n^{0.6}/2\right) + \exp(-\Omega(n)) \\
&\leq 2\exp\left(-\frac{n^{1.2}/4}{2N(4r_{\max})^2}\right) + \exp(-\Omega(n)) = o(\exp(-n^{0.1})).
\end{aligned}$$

According to the computation in the proof of Theorem 36 and using the fact that $E(T_{1/3}) = E(T_{\tilde{A}^*_\varepsilon}) - nH_{n/3}$, we obtain with probability $1 - O(\exp(-n^{0.1}))$ that $0.2549n \leq T_{1/3} \leq 0.2675n$.

Consider a budget of $B = kn$ iterations with $k \geq 0.2675$. Let $s := \lfloor 0.2675n \rfloor$. With probability $1 - O(\exp(-n^{-0.1}))$, a run of algorithm $\tilde{A}^*_\varepsilon$ has $X_s \leq n/3$. Conditional on this, in the remainder $\tilde{A}^*_\varepsilon$ mutates exactly one bit in each iteration according to Lemma 32. Since $E(X_t \mid X_{t-1}) = X_{t-1}(1 - 1/n)$ in this case, we have for all $t \geq s$ that

$$E(X_t \mid X_s) = E(X_s)\,(1 - 1/n)^{t-s} \leq (n/3)(1 - 1/n)^{t-s}.$$

Therefore, with a budget of $B \geq 0.2675n$ algorithm $\tilde{A}^*_\varepsilon$ reaches a fitness distance X_B satisfying

$$E(X_B) \leq E(X_B \mid X_s \leq n/3) + n \cdot \Pr(X_s > n/3) \leq (1 + o(1))(n/3)(1 - 1/n)^{B-0.2675n}.$$

Using the same reasoning for RLS, we compute for Y_B the fitness distance RLS reaches with the same budget of B that

$$\begin{aligned}
E(Y_B) &= (n/2)(1 - 1/n)^B \\
&= (3/2)(1 - 1/n)^{0.2675n}(n/3)(1 - 1/n)^{B-0.2675n} \\
&\geq (1 - o(1))(3/2)\exp(-0.2675)E(X_B) \\
&= (1 - o(1))1.1479...\,E(X_B).
\end{aligned}$$

In other words, $E(X_B) \leq (1 + o(1))0.8711...\,E(X_A)$, that is, with the same budget, Algorithm $\tilde{A}^*_\varepsilon$ is roughly 13% closer to the optimum than RLS.

Acknowledgments This research benefited from the support of the "FMJH Program Gaspard Monge in optimization and operation research", and from the support to this program from Électricité de France. It has also been supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH.

References

[AD18] Denis Antipov and Benjamin Doerr. Precise Runtime Analysis for Plateaus. In *Proc. of Parallel Problem Solving From Nature (PPSN'18)*, volume 11102 of *Lecture Notes in Computer Science*, pages 117–128. Springer, 2018.

[Azu67] Kazuoki Azuma. Weighted sums of certain dependent variables. *Tohoku Mathematical Journal*, 19:357–367, 1967.

[Bäc92] Thomas Bäck. The interaction of mutation rate, selection, and self-adaptation within a genetic algorithm. In *Proc. of Parallel Problem Solving from Nature (PPSN'92)*, pages 87–96. Elsevier, 1992.

[BBD+09] Surender Baswana, Somenath Biswas, Benjamin Doerr, Tobias Friedrich, Piyush P. Kurur, and Frank Neumann. Computing single source shortest paths using single-objective fitness functions. In *Proc. of the 10th ACM Workshop on Foundations of Genetic Algorithms (FOGA'09)*, pages 59–66. ACM, 2009.

[BD18] Nathan Buskulic and Carola Doerr. personal communication, 2018.

[BDK16] Maxim Buzdalov, Benjamin Doerr, and Mikhail Kever. The unrestricted black-box complexity of jump functions. *Evolutionary Computation*, 24:719–744, 2016.

[BDN10] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Proc. of Parallel Problem Solving from Nature (PPSN'10)*, volume 6238 of *Lecture Notes in Computer Science*, pages 1–10. Springer, 2010.

[DD16] Benjamin Doerr and Carola Doerr. The impact of random initialization on the runtime of randomized search heuristics. *Algorithmica*, 75:529–553, 2016.

[DD18] Benjamin Doerr and Carola Doerr. Theory of parameter control mechanisms for discrete black-box optimization: Provable performance gains through dynamic parameter choices. In Benjamin Doerr and Frank Neumann, editors, *Theory of Randomized Search Heuristics in Discrete Search Spaces*. Springer, 2018. To appear.

[DDE15] Benjamin Doerr, Carola Doerr, and Franziska Ebel. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science*, 567:87–104, 2015.

[DDY16a] Benjamin Doerr, Carola Doerr, and Jing Yang. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proc. of Parallel Problem Solving from Nature (PPSN'16)*, volume 9921 of *Lecture Notes in Computer Science*, pages 824–834. Springer, 2016.

[DDY16b] Benjamin Doerr, Carola Doerr, and Jing Yang. Optimal parameter choices via precise black-box analysis. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'16)*, pages 1123–1130. ACM, 2016.

[DFW10] Benjamin Doerr, Mahmoud Fouz, and Carsten Witt. Quasirandom evolutionary algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'10)*, pages 1457–1464. ACM, 2010.

[DFW11] Benjamin Doerr, Mahmoud Fouz, and Carsten Witt. Sharp bounds by probability-generating functions and variable drift. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'11)*, pages 2083–2090. ACM, 2011.

[DGWY17] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. The (1+λ) evolutionary algorithm with self-adjusting mutation rate. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*, pages 1351–1358. ACM, 2017. Full version available at http://arxiv.org/abs/1704.02191.

[DJS+13] Benjamin Doerr, Thomas Jansen, Dirk Sudholt, Carola Winzen, and Christine Zarges. Mutation rate matters even when optimizing monotone functions. *Evolutionary Computation*, 21:1–21, 2013.

[DJW02] Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81, 2002.

[DJW06] Stefan Droste, Thomas Jansen, and Ingo Wegener. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems*, 39:525–544, 2006.

[DJW12] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. Multiplicative drift analysis. *Algorithmica*, 64:673–697, 2012.

[DJWZ13] Benjamin Doerr, Thomas Jansen, Carsten Witt, and Christine Zarges. A method to derive fixed budget results from expected optimisation times. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'13)*, pages 1581–1588. ACM, 2013.

[DKLW13] Benjamin Doerr, Timo Kötzing, Johannes Lengler, and Carola Winzen. Black-box complexities of combinatorial problems. *Theoretical Computer Science*, 471:84–106, 2013.

[DKV13] Benjamin Doerr, Bojana Kodric, and Marco Voigt. Lower bounds for the runtime of a global multi-objective evolutionary algorithm. In *Proc. of the Congress on Evolutionary Computation (CEC'13)*, pages 432–439. IEEE, 2013.

[DLMN17] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*, pages 777–784. ACM, 2017. Full version available at http://arxiv.org/abs/1703.03334.

[DLOW18] Benjamin Doerr, Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*, pages 1015–1022. ACM, 2018.

[DNDD+18] Raphaël Dang-Nhu, Thibault Dardinier, Benjamin Doerr, Gautier Izacard, and Dorian Nogneng. A new analysis method for evolutionary optimization of dynamic and noisy objective functions. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*, pages 1467–1474. ACM, 2018.

[Doe11] Benjamin Doerr. Analyzing randomized search heuristics: Tools from probability theory. In Anne Auger and Benjamin Doerr, editors, *Theory of Randomized Search Heuristics*, pages 1–20. World Scientific Publishing, 2011.

[Doe18a] Benjamin Doerr. Better runtime guarantees via stochastic domination. In *Proc. of Evolutionary Computation in Combinatorial Optimization (EvoCOP'18)*, pages 1–17. Springer, 2018. Full version available at http://arxiv.org/abs/1801.04487.

[Doe18b] Carola Doerr. Complexity theory for discrete black-box optimization heuristics. *CoRR*, abs/1801.02037, 2018. To appear in the book "Theory of Randomized Search Heuristics in Discrete Search Spaces".

[dPdLDD15] Axel de Perthuis de Laillevault, Benjamin Doerr, and Carola Doerr. Money for nothing: Speeding up evolutionary algorithms through better initialization. In *Proc. of the Genetic and Evolutionary Computation Conference (GECCO'15)*, pages 815–822. ACM, 2015.

[DW12] Benjamin Doerr and Carola Winzen. Memory-restricted black-box complexity of OneMax. *Information Processing Letters*, 112:32–34, 2012.

[DW14] Benjamin Doerr and Carola Winzen. Ranking-based black-box complexity. *Algorithmica*, 68:571–609, 2014.

[FCSS08] Álvaro Fialho, Luís Da Costa, Marc Schoenauer, and Michèle Sebag. Extreme value based adaptive operator selection. In *Proc. of Parallel Problem Solving from Nature (PPSN'08)*, volume 5199 of *Lecture Notes in Computer Science*, pages 175–184. Springer, 2008.

[FCSS09] Álvaro Fialho, Luís Da Costa, Marc Schoenauer, and Michèle Sebag. Dynamic multi-armed bandits and extreme value-based rewards for adaptive operator selection in evolutionary algorithms. In *Proc. of Learning and Intelligent Optimization (LION'09)*, volume 5851 of *Lecture Notes in Computer Science*, pages 176–190. Springer, 2009.

[FT11] Hervé Fournier and Olivier Teytaud. Lower bounds for comparison based evolution strategies using vc-dimension and sign patterns. *Algorithmica*, 59:387–408, 2011.

[GKS99] Josselin Garnier, Leila Kallel, and Marc Schoenauer. Rigorous hitting times for binary mutations. *Evolutionary Computation*, 7:173–203, 1999.

[GW15] Christian Gießen and Carsten Witt. Population size vs. mutation strength for the (1+λ) EA on OneMax. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'15)*, pages 1439–1446. ACM, 2015.

[GW16] Christian Gießen and Carsten Witt. Optimal mutation rates for the (1+λ) EA on OneMax. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'16)*. ACM, 2016.

[HPR+18] Hsien-Kuei Hwang, Alois Panholzer, Nicolas Rolin, Tsung-Hsi Tsai, and Wei-Mei Chen. Probabilistic analysis of the (1+1)-evolutionary algorithm. *Evolutionary Computation*, 26:299–345, 2018.

[HY04] Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3:21–35, 2004.

[Jäg08] Jens Jägersküpper. A blend of Markov-chain and drift analysis. In *Proc. of Parallel Problem Solving from Nature (PPSN'08)*, volume 5199 of *Lecture Notes in Computer Science*, pages 41–51. Springer, 2008.

[Joh10] Daniel Johannsen. *Random combinatorial structures and randomized search heuristics*. PhD thesis, Saarland University, 2010.

[JZ14] Thomas Jansen and Christine Zarges. Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science*, 545:39–58, 2014.

[Len18] Johannes Lengler. A general dichotomy of evolutionary algorithms on monotone functions. In *Proc. of Parallel Problem Solving from Nature (PPSN'18)*, volume 11102 of *Lecture Notes in Computer Science*, pages 3–15. Springer, 2018. Full version available at http://arxiv.org/abs/1803.09227.

[LOW17] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. On the runtime analysis of generalised selection hyper-heuristics for pseudo-Boolean optimisation. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*, pages 849–856. ACM, 2017.

[LS18] Johannes Lengler and Angelika Steger. Drift analysis and evolutionary algorithms revisited. *Combinatorics, Probability & Computing*, 27:643–666, 2018.

[LW12] Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[MRC09] Boris Mitavskiy, Jonathan E. Rowe, and Chris Cannings. Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *Journal of Intelligent Computing and Cybernetics*, 2:243–284, 2009.

[RS14] Jonathan E. Rowe and Dirk Sudholt. The choice of the offspring population size in the (1,λ) evolutionary algorithm. *Theoretical Computer Science*, 545:20–38, 2014.

[Sud13] Dirk Sudholt. A new method for lower bounds on the running time of evolutionary algorithms. *IEEE Transactions on Evolutionary Computation*, 17:418–435, 2013.

[Wit13] Carsten Witt. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability & Computing*, 22:294–318, 2013.

[Wit14] Carsten Witt. Revised analysis of the (1+1) EA for the minimum spanning tree problem. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'14)*, pages 509–516. ACM, 2014.