Tight Runtime Bounds for Static Unary Unbiased Evolutionary Algorithms on Linear Functions

Carola Doerr, Duri Andrea Janett, Johannes Lengler

Tight Runtime Bounds for Static Unary Unbiased Evolutionary Algorithms on Linear Functions

Carola Doerr
Sorbonne Université, CNRS, LIP6
Paris, France

Duri Andrea Janett*
Department of Computer Science,
ETH Zürich
Zürich, Switzerland

Johannes Lengler
Department of Computer Science,
ETH Zürich
Zürich, Switzerland

ABSTRACT

In a seminal paper in 2013, Witt showed that the (1+1) Evolutionary Algorithm with standard bit mutation needs time $(1+o(1))n \ln n/p_1$ to find the optimum of any linear function, as long as the probability p_1 to flip exactly one bit is $\Theta(1)$. In this paper we investigate how this result generalizes if standard bit mutation is replaced by an arbitrary unbiased mutation operator. This situation is notably different, since the stochastic domination argument used for the lower bound by Witt no longer holds. In particular, starting closer to the optimum is not necessarily an advantage, and OneMax is no longer the easiest function for arbitrary starting positions.

Nevertheless, we show that Witt's result carries over if p_1 is not too small, with different constraints for upper and lower bounds, and if the number of flipped bits has bounded expectation χ. Notably, this includes some of the heavy-tail mutation operators used in fast genetic algorithms, but not all of them. We also give examples showing that algorithms with unbounded χ have qualitatively different trajectories close to the optimum.

CCS CONCEPTS

• **Theory of computation → Theory of randomized search heuristics**.

KEYWORDS

Runtime analysis, Theory of Evolutionary Computation, Mutation Operators

1 INTRODUCTION

One of the most crucial ingredients of evolutionary algorithms is the *mutation operator*, i.e., the procedure that describes how to generate offspring from a single parent. On the hypercube $\{0,1\}^n$, for a long time the undisputed default was to use *standard bit mutation*, which flips each bit of the parent independently with the same probability. However, this convention has been challenged in the last years; for example via the *fast* mutation operators [27], for which the number of flipped bits follows a heavy-tailed distribution. The advantages of using heavy-tailed distributions are rather impressive [28]. They are slightly worse for hillclimbing, but the runtime deteriorates only by a constant factor that can be chosen close to one. However, they are massively better at escaping local optima. While it takes $e^{\Omega(k \log k)}$ steps to make a jump of size k with standard bit mutation of mutation rate $\Theta(1/n)$, it only takes $k^{O(1)}$ steps with fast mutation operators. Consequently, they are faster on landscapes with local optima, like the Jump function [3, 27] and its generalizations [5–7], or random MAX-3SAT instances [2].[1] Heavy-tailed distributions can also help on unimodal landscapes like OneMax. For example, the $(1 + (\lambda, \lambda))$ GA [18] was shown to achieve linear expected runtime [2] when equipped with fast mutation operators, which is asymptotically best possible.

Other benchmarks on which fast mutation operators or other unbiased mutation operators than standard bit mutation have been found to be useful include theoretical benchmarks like LeadingOnes [56] and TwoMax [38], network problems like maximum cut [37, 38, 50], minimum vertex cover [10, 38], maximum independent set [57], maximum flow [45] and SAT [51], landscape classes like submodular functions [37, 50] and random NK-landscapes [57], multi-objective settings [22, 29, 32] and other problems like subset selection [55], the N-queens problem [57], the symmetric mutual information problem [37, 50] and many more [1, 21, 40, 47, 48, 57]. Such mutation operators are integrated into standard benchmarking tools like the IOHprofiler [35] and Nevergrad [8], and they have been used as building blocks for more sophisticated algorithms [13, 14, 30, 46, 49].

The large success of non-standard mutation operators raises the desire to analyze which operators are (provably) optimal for a given problem setting. Such questions can be answered in the black-box complexity framework proposed in [36] (see [33] for a survey on the role of black-box complexity for evolutionary computation). Particularly interesting for the study of mutation operators is the unary unbiased black-box complexity model defined in [42]. Unary unbiased black-box algorithms create solution candidates by sampling uniformly at random (u.a.r.) or by selecting one previously evaluated point x and a search radius r (both possibly random) and then sampling the solution candidate u.a.r. among all points at Hamming distance r from x. The unary unbiased black-box complexity of a collection $\mathcal{F}$ of functions is then the best (over all unary unbiased algorithms) worst-case (over all problem instances in $\mathcal{F}$) expected runtime. The study of unary unbiased black-box complexities has led to important insights into the limitation of mutation-based algorithms [17, 19, 20, 23, 26, 31, 41, 42], which were exploited for the design of faster algorithms such as the $(1 + (\lambda, \lambda))$ GA in [18].

For OneMax, a tight bound for the unary unbiased black-box complexity was proven in [20]. It was shown there that the *drift-maximizing* algorithm that at every step chooses the mutation operator that maximizes the expected progress achieves asymptotically optimal expected runtime, up to small lower order terms. Zooming further into this problem for concrete dimensions, Buskulic and Doerr [9] showed that slightly better performance can be achieved

*Also with Sorbonne Université, CNRS, LIP6, Paris, France.

[1] When the required jump size k is known in advance, then choosing the mutation rate to be k/n is optimal, as shown in [27]. The advantage of the fast mutation operator is that k does not need to be known.

by increasing the mutation rates, i.e., by implementing a more risky strategy that, at several stages that are sufficiently far away from the optimum, flips more bits (in the hope of making more progress and at the cost of a smaller success probability). The approach developed by [9] was later extended in [11] to compute the optimal mutation rates for the $(1+1)$-EA and the $(1+\lambda)$-EA optimizing OneMax. The best *static* unary unbiased mutation operator for the $(1+\lambda)$-EA for a number of different combinations of n and λ was numerically approximated in [12]. In particular, it was shown there that the optimal mutation operators are none of the standard choices that are typically used in evolutionary algorithms. These results demonstrate that even for the optimization of OneMax our understanding of optimal mutation operators is rather limited, both in the static and in the dynamic case.

Our Results: We aim to extend in this work the above-mentioned results to the optimization of a larger class of functions. The first natural extension of OneMax are linear functions, so we primarily focus on these. Our particular aim is to derive tight bounds for the expected runtime of the $(1+1)$-EA equipped with an arbitrary unary unbiased mutation operator.

To express our main result, we briefly recall from [33] that every unbiased mutation operator can be described by a sequence of $n+1$ probabilities $p_0, p_1, \dots, p_n$ that sum up to one. We thus identify the mutation operator with the sequence $\mathcal{D} = (p_0, p_1, \dots, p_n)$ and write $(1+1)\text{-EA}_{\mathcal{D}}$ for the $(1+1)$-EA that generates its solution candidates using the mutation operator $\text{mut}_{\mathcal{D}}$ that first draws an index $i \in [0, n]$ according to the probabilities (i.e., it picks i with probability p_i), and then flips a uniformly random set of exactly i bits. Every $(1+1)$-EA equipped with an arbitrary but static unary unbiased mutation operator can be expressed as a $(1+1)\text{-EA}_{\mathcal{D}}$. We show the following.

Theorem 1.1. *Consider the $(1+1)$-EA$_{\mathcal{D}}$ for a distribution $\mathcal{D} = (p_0, p_1, \dots, p_n)$ with mean χ. If $p_1 = \Theta(1)$ and $\chi = O(1)$, then the expected runtime on any linear function on $\{0,1\}^n$ with non-zero weights is*

$$(1 \pm o(1)) \frac{1}{p_1} \cdot n \ln n. \tag{1}$$

More precise versions of Theorem 1.1 will be presented in Corollary 3.3 and Theorem 4.1. In particular, we will show that the lower bound holds for any function with unique global optimum if $p_{n-1} = o(p_1)$. Moreover, the conditions on p_1 and χ in Theorem 1.1 can be slightly relaxed. We show that the runtime remains unchanged if $\chi^3 p_1^{-2}(1-p_0)^{-1} = o(\ln n / \ln\ln n)$, which is probably not tight. However, we also show that the condition is not superfluous either. If p_1 becomes too small, or χ becomes too large, then the behavior of the algorithm starts to change, see Section 3.1.

Theorem 1.1 can be seen as a generalization of Witt's seminal work [53] on linear functions, where he showed that the expected runtime of the the $(1+1)$-EA using standard bit mutation with arbitrary mutation rates c/n is $(1 \pm o(1)) \frac{e^c}{c} n \ln n = (1 \pm o(1)) \frac{1}{p_1} n \ln n$, where p_1 is the probability that the mutation flips a single bit. Our proof of the upper bounds closely follows his, but we need to adapt his potential function to account for the fact that the probabilities p_i may follow any distribution.

For the lower bound we follow the proof strategy from [20]. In particular, we use the same symmetrized OneMax potential $X_t = \min_x \min\{\text{Om}(x), n - \text{Om}(x)\}$, where $\text{Om}(x)$ is the number of one-bits in x and the minimum goes over all previously visited search points x. We show that for a wide range of values of X_t, the drift is maximized either by single-bit flips or by $(n-1)$-bit flips, and with a parent that achieves the minimum in X_t. This allows us to compute an upper bound on the drift, and to use the variable drift lower bound from [20]. We obtain a lower bound for any function with unique local optimum, but then p_1 needs to be replaced by $p_1 + p_{n-1}$ in (1). This is not an artifact of our proof, since we give examples showing that the dependence on p_{n-1} is real.

Finally, we also show (Section 4.2) that stochastic domination no longer applies when standard bit mutation is replaced by other unary unbiased mutation operators, in the sense that starting closer to the optimum can increase the runtime asymptotically. This even holds on OneMax. As a consequence, non-elitist algorithms may be faster than elitist algorithms on OneMax.

Other Related Work. Apart from black-box complexities, only few things are known in general about the class of unbiased mutation operators. Antipov and Doerr [4] investigated the mixing time on plateaus for the $(1+1)$-EA with arbitrary unbiased mutation operator. Lengler [43] studied the $(1+1)$-EA, the $(1+\lambda)$-EA, the $(\mu+1)$-EA and the $(\mu+1)$-GA with arbitrary unbiased mutation operators on monotone functions. He found that those algorithms can optimize all monotone functions if the second moment of the number of bit flips is small compared to the first moment, but that they need exponential time on HotTopic functions otherwise. In particular, all heavy-tail distributions lead to exponential runtimes on HotTopic.

Full proofs can be found in the arXiv version of the paper [34].

2 ELITIST (1+1) UNARY UNBIASED EAS

We use the following notation. For $a, b \in \mathbb{N}$ with $a \le b$ we write $[a, b] = \{a, a+1, \dots, b\}$ and $[b] = [1, b] = \{1, \dots, b\}$. We write a vector $x \in \{0,1\}^n$ as $x = (x_1, \dots, x_n)$. The *OneMax value* $\text{Om}(x) := \sum_{i=1}^n x_i$ of x is the number of one-bits in x. We write $\vec{0}$ and $\vec{1}$ for the vectors in $\{0,1\}^n$ with $\text{Om}(\vec{0}) = 0$ and $\text{Om}(\vec{1}) = n$, respectively. *With high probability (w.h.p.)* means with probability $1 - o(1)$ as $n \to \infty$.

We identify probability distributions $\mathcal{D}$ on $[0, n]$ with sequences $(p_0, p_1, \dots, p_n)$ such that $p_i \ge 0$ for all $i \in [0, n]$ and $\sum_{i \in [0,n]} p_i = 1$, where the probability of obtaining i from $\mathcal{D}$ is p_i. We associate to any such distribution $\mathcal{D}$ the mutation operator $\text{mut}_{\mathcal{D}}$ which draws k from $\mathcal{D}$ and then applies the flip_k operator which flips a uniform random set of exactly k positions. The probability that $\text{mut}_{\mathcal{D}}$ flips the i-th bit equals χ/n, where χ is the expected value of $\mathcal{D}$.

For a probability distribution $\mathcal{D}$ on $[0, n]$, we define the $(1+1)$-EA$_{\mathcal{D}}$ as the elitist $(1+1)$ algorithm which uses $\text{mut}_{\mathcal{D}}$ as mutation operator, see Algorithm 1. Its *runtime* on a function f is the number of fitness evaluations before it finds a global optimum. Following the discussion in [20, 33], the class of elitist (1+1) unary unbiased black-box algorithms with static mutation operators coincides exactly with the collection of all $(1+1)\text{-EA}_{\mathcal{D}}$ with $\mathcal{D}$ as above.

With *linear functions* we always refer to functions $f : \{0,1\}^n \to \mathbb{R}$; $f(x) = \sum_{i=1}^n w_i x_i$ for weights w_i. By unbiasedness of the $(1+1)$-EA$_{\mathcal{D}}$, we may assume that the weights are positive and sorted, $0 < w_1 \le \dots \le w_n$.

Algorithm 1: The $(1+1)$-EA$_{\mathcal{D}}$ for a fixed distribution $\mathcal{D}$ and maximizing a function $f : \{0,1\}^n \to \mathbb{R}$.

```
Sample x from {0,1}^n uniformly at random;
for t = 0, 1, 2, 3, ... do
    Sample k ~ D;
    Create y ← flip_k(x);
    if f(y) ≥ f(x) then x ← y;
```

3 UPPER BOUNDS AND TIGHTNESS RESULTS

We first note the following, simpler version of the upper bound stated in Theorem 1.1 for OneMax, which does not require any assumption on the distribution $\mathcal{D}$. It straightforwardly follows from the standard multiplicative drift theorem [25], applied to the lower bound on the drift obtained by considering only 1-bit flips.

Theorem 3.1. *Let $\mathcal{D} = (p_0, p_1, \dots, p_n)$ be a probability distribution on $[0, n]$. The runtime of the $(1+1)$-EA$_{\mathcal{D}}$ on OneMax is at most*

$$(1 \pm o(1)) \frac{1}{p_1} n \ln n \tag{2}$$

in expectation and with high probability.

The key ingredient for generalizing the bound from OneMax to all linear functions as in Theorem 1.1 is the following theorem, which generalizes [53, Theorem 4.1] to the $(1+1)$-EA$_{\mathcal{D}}$ with (almost) arbitrary $\mathcal{D}$. Our proof follows [53], with the following differences: First, we noted a mistake in the proof of the upper bound in [53]. Equation (4.2) there does not hold for the events A_i as defined in [53]. We thank Carsten Witt for providing the following fix upon our inquiry (personal communication): By conditioning the events A_i on the event that the offspring is accepted, equation (4.2) holds as in that case, the expectation is zero if none of the A_i occur. Furthermore, the inequality (4.3) in [53] still holds, which can be shown by applying Bayes' theorem and linearity of expectation. Details can be found in the arXiv version of this paper [34].

Apart from this issue, the biggest challenge was to adapt the potential function used in [53], since we need to deal with arbitrary unbiased mutation operators. In particular, our potential involves the quantities χ and p_1. With the modified potential, we can show the following generalization of [53, Theorem 4.1].

Theorem 3.2. *Let $\mathcal{D} = (p_0, p_1, \dots, p_n)$ be a probability distribution on $[0, n]$ with expectation χ and with $p_1 > 0$. Then the runtime of the $(1+1)$-EA$_{\mathcal{D}}$ on any linear function on n variables is at most*

$$b(r) \coloneqq \frac{n}{p_1} \cdot \frac{\alpha}{\alpha - 1} \cdot \left(\frac{\alpha n \chi^3}{(n-1)p_1^2} + \ln\left(\frac{(n-1)p_1^2}{\chi^3} \right) + r \right) \tag{3}$$

with probability at least $1 - e^{-r}$ for any $r > 0$, and it is at most $b(1)$ in expectation, where $\alpha > 1$ can be chosen arbitrarily.

Proof Sketch for Theorem 3.2. The proof works by applying the multiplicative drift theorem to a carefully chosen potential. To this end, following [53], we define a new (linear) function g, and consider the stochastic process $X_t = g(x^{(t)})$, where $x^{(t)}$ is the current search point of the $(1+1)$-EA$_{\mathcal{D}}$ at time t. The weights g_i of the function g are as follows. For all $1 \le i \le n$, we let

$$\gamma_i \coloneqq \left(1 + \frac{\alpha \chi^3}{(n-1)p_1^2}\right)^{i-1}, \tag{4}$$

put $g_1 \coloneqq \gamma_1 = 1$, and for $2 \le i \le n$, we set

$$g_i \coloneqq \min\left\{ \gamma_i, g_{i-1} \frac{w_i}{w_{i-1}} \right\} \ge 1. \tag{5}$$

We show a lower bound on the expected change in the potential of

$$\mathbb{E}\left[X_t - X_{t+1} \mid X_t = s\right] \ge s \cdot \frac{p_1}{n} \cdot \frac{\alpha - 1}{\alpha}, \tag{6}$$

following the strategy of [53], i.e., by defining suitable events A_i, showing that (4.2) and (4.3) from [53] hold, and using the definition of γ_i. The result then follows by the multiplicative drift theorem. □

From Theorem 3.2 we obtain the following upper bound, which relaxes the conditions on p_1 and χ in Theorem 1.1 and shows that the bound holds not only in expectation but also w.h.p.

Corollary 3.3. *Let $\mathcal{D} = (p_0, p_1, \dots, p_n)$ be a probability distribution on $[0, n]$ with expectation χ. Assume that $p_1 > 0$ and $\chi^3 p_1^{-2} (1 - p_0)^{-1} = o(\ln n / \ln \ln n)$. Then the runtime of the $(1+1)$-EA$_{\mathcal{D}}$ on any linear function is at most*

$$(1 + o(1)) \frac{1}{p_1} \cdot n \ln n \tag{7}$$

in expectation and with high probability.

Note that since $1 - p_0 \ge p_1$, we could replace the requirement $\chi^3 p_1^{-2} (1 - p_0)^{-1} = o(\ln n / \ln \ln n)$ by the stronger requirement $\chi^3 / p_1^3 = o(\ln n / \ln \ln n)$. In particular, this is trivially satisfied if $p_1 = \Theta(1)$ and $\chi = O(1)$, as required in Theorem 1.1.

Proof Sketch for Corollary 3.3. We first treat the case $p_0 = 0$, which implies $\chi \ge 1$. Let $\alpha \coloneqq \ln \ln n$. As in [53], $\alpha/(\alpha - 1) = 1 + O(1/\ln \ln n)$, and $\alpha^2/(\alpha - 1) = O(\ln \ln n)$. Thus $b(r)$ in Theorem 3.2 is at most

$$\frac{n}{p_1} \left(o(\ln n) + (1 + o(1)) \left(\ln n + r \right) \right) \tag{8}$$

Taking $r \coloneqq \ln \ln n$ and $r \coloneqq 1$, the corollary follows for $p_0 = 0$.

For $p_0 > 0$, we define an auxiliary distribution $\mathcal{D}' = (p'_0, \dots, p'_n)$ with $p'_0 = 0$, which is essentially $\mathcal{D}$ conditioned on not choosing 0. Then we show that the already proven case of Corollary 3.3 is applicable to $\mathcal{D}'$. Thus, the runtime of the $(1+1)-EA_{\mathcal{D}'}$ is at most

$$(1 + o(1)) \frac{1}{p'_1} \cdot n \ln n = (1 + o(1)) \frac{1 - p_0}{p_1} \cdot n \ln n, \tag{9}$$

in expectation and with high probability. Since no-bit flips are just idle steps of the $(1+1)$-EA$_{\mathcal{D}}$, the $(1+1)$-EA$_{\mathcal{D}}$ and the $(1+1)$-EA$_{\mathcal{D}'}$ follow the same trajectory through the search space, except that the $(1+1)$-EA$_{\mathcal{D}}$ needs to wait for $1/(1-p_0)$ in expectation for a non-idle step. Hence, the $(1+1)$-EA$_{\mathcal{D}}$ is by a factor of $1/(1-p_0)$ slower than the $(1+1)$-EA$_{\mathcal{D}'}$, and the claim follows from (9). We omit the details. □

Under some less restrictive conditions, we can give a polynomial upper bound on the runtime.

Corollary 3.4. *Let $\mathcal{D} = (p_0, p_1, \dots, p_n)$ be a probability distribution on $[0, n]$ with expectation χ. The runtime of the $(1+1)$-EA$_\mathcal{D}$ on any linear function is $O(n^4/p_1^3)$ in expectation and with high probability. In particular, it is polynomial if $1/p_1$ is polynomial in n.*

Proof. We take $\alpha := 2$, and apply Theorem 3.2. Noting that $\chi \geq p_1$ and thus $(n-1)p_1^2/\chi^3 \leq n/p_1$, by (3) we can bound the runtime of the $(1+1)$-EA$_\mathcal{D}$ on any linear function by

$$\frac{2n}{p_1}\left(\frac{2n\chi^3}{(n-1)p_1^2} + \ln\left(\frac{n}{p_1}\right) + r\right) = O\left(\frac{n^4}{p_1^3}\right), \tag{10}$$

where we use $\chi \leq n$ and pick $r = \ln n$ or $r = 1$ for the last step. □

3.1 Tightness

We now discuss that some requirements on p_1 and χ are necessary.

Requirement on p_1. We start with a proposition saying that the leading constant can change if $p_1 = n^{-c}$ for any $c > 0$. In fact, this is already the case for OneMax, as the following example shows.

Proposition 3.5. *Let $0 < c < 1$ be constant. Consider the $(1+1)$-EA$_\mathcal{D}$ with distribution $\mathcal{D} = (p_0, p_1, \dots, p_n)$ defined by $p_1 = n^{-c}$ and $p_2 = 1 - p_1$. Then there is $\varepsilon > 0$ such that for sufficiently large n the expected runtime on OneMax is at most*

$$(1-\varepsilon)\cdot\frac{1}{p_1}\cdot n\ln n.$$

Hence we have a strictly smaller leading constant than in Corollary 3.3. The reason for this effect is that if $p_1 = n^{-\Omega(1)}$, for Hamming distances in the range $[n^{1-c/2}, n]$ from the optimum, two-bit flips are more effective than single-bit flips. This range is thus traversed more quickly. With single-bit flips, the algorithm would need time $\Omega(n\ln n/p_1)$ to traverse this region, but it can be traversed in time $o(n\ln n/p_1)$ by two-bit flips. Hence, the time spent in this phase becomes negligible. Even though this region is still far away from the optimum, it consumes a constant fraction of the total runtime if the algorithm is restricted to single-bit flips. Hence, the speed-up from two-bit flips eliminates this constant fraction from the total runtime, and thus reduces the leading constant of the total runtime. This shows that the lower bound in Theorem 1.1 can not hold if $p_1 = n^{-\Omega(1)}$. On the other hand, we will show that it does hold for all $p_1 = n^{-o(1)}$, which is tight by the above discussion.

Requirement on χ. Other than for p_1, we could not derive a statement about the runtime, but the following proposition shows that, close to the optimum, the behavior of the algorithm changes substantially if χ is large. Recall that from any parent at distance one from the optimum, we have a probability of $p_1\cdot 1/n$ to create the optimum as offspring. Hence, one would naively expect to wait at most for n/p_1 rounds in expectation to find the optimum. However, this is wrong for large values of χ, as the following proposition shows.

Proposition 3.6. *Let x be a search point at Hamming distance one from the optimum $\vec{1}$. Let $\mathcal{D}$ be any probability distribution on $[0, n]$ with mean χ. For a linear function f, let $T^x_\mathcal{D}(f)$ be the number of iterations until the $(1+1)$-EA$_\mathcal{D}$ with starting position x finds the optimum.*

(a) There is a linear function f depending on x such that $\mathbb{E}[T^x_\mathcal{D}(f)] = \Omega(n\ln\chi)$.

(b) If $\chi = \omega(1)$ then there is a linear function f depending on x such that $T^x_\mathcal{D}(f) = \omega(n)$ with high probability.

(c) If $\chi = O(1)$ and $p_1 = \Theta(1)$ then $\mathbb{E}[T^x_\mathcal{D}(f)] = O(n)$ for every linear function f.

Proof. *(a) and (b).* It is clear that the number of iterations is at least $\Omega(n)$, so we may assume $\chi \geq 4$. Since x has Hamming distance one from $\vec{1}$, it differs in exactly one position from $\vec{1}$. We may assume that this is the first position. Then we define f via $f(x) := n\cdot x_1 + \sum_{i=2}^n x_i$, i.e, we give weight n to the first position and weight 1 to all other positions. When in x, the algorithm will accept any offspring that flips the first position. Let us call R the number of bits that are flipped in this mutation. Then we may compute the distribution of R via Bayes formula as

$$\Pr[R = r] = \frac{\Pr[R = r \text{ and pos. 1 flipped}]}{\sum_{s\in[n]}\Pr[R = s \text{ and pos. 1 flipped}]}. \tag{11}$$

Note that $\Pr[R = s \text{ and pos. 1 flipped}] = \Pr[R = s]\cdot\Pr[\text{pos 1 flipped} \mid R = s] = p_s\cdot s/n$, where the conditional probability is s/n since the mutation operator is unbiased. Hence, (11) simplifies to

$$\Pr[R = r] = \frac{p_r\cdot r/n}{\sum_{s\in[n]} p_s\cdot s/n} = \frac{p_r\cdot r}{\chi}. \tag{12}$$

In particular, this implies for every $\gamma \leq 1$,

$$\Pr[R \leq \gamma\chi] = \sum_{r=1}^{\lfloor\gamma\chi\rfloor}\frac{p_r\cdot r}{\chi} \leq \frac{\gamma\chi\sum_{r=1}^{\lfloor\gamma\chi\rfloor} p_r}{\chi} \leq \gamma.$$

Hence, with probability at least $1-\gamma$, the $(1+1)$-EA$_\mathcal{D}$ proceeds from x to a search point in Hamming distance at least $\gamma\chi - 1$ from $\vec{1}$.

For claim *(a)* we set $\gamma := 1/2$ and obtain a search point in Hamming distance at least $\chi/2 - 1 \geq \chi/4$. Once this search point is reached, the algorithm does not accept any mutation which flips position 1 again, since this would decrease the fitness. Hence, the algorithm simply has to solve OneMax on the remaining $n-1$ bits. In expectation, this takes time $\Omega(n\ln\chi)$ since the unbiased black-box complexity for solving OneMax from a starting point in distance at least $\chi/4$ is $\Omega(n\ln\chi)$ (implicit in [42]). Since this case happens with probability at least $1-\gamma = 1/2$, we obtain

$$\mathbb{E}[T^x_\mathcal{D}(f)] \geq \tfrac{1}{2}\cdot\Omega(n\ln\chi) = \Omega(n\ln\chi).$$

For claim *(b)*, we choose $\gamma := \chi^{-1/2}$. Then $1-\gamma = 1-o(1)$, so with high probability the $(1+1)$-EA$_\mathcal{D}$ proceeds from x to a search point in distance at least $d := \gamma\chi - 1 = \chi^{1/2} - 1 = \omega(1)$ from $\vec{1}$. As for part (a), from this point onwards the algorithm needs to solve OneMax on $n-1$ bits. Thus the algorithm needs to traverse the interval from $d' := \min\{d, n^{1/3}\}$ to the optimum. We show that w.h.p. this takes time at least t_0 for some $t_0 = \Omega(n\ln d') = \omega(n)$. It can be shown that the probability to find an improvement with r-bit flips for any $r \geq 2$ is $O((d'/n))^2 \leq n^{-4/3}$. Hence, in time t_0 the expected number of such improvements is $O(t_0 n^{-4/3}) = o(1)$, and by Markov's inequality no r-bit flip finds an improvement for $r \geq 2$. Hence, we may pessimistically assume that the algorithm only uses single-bit flips, i.e., that it is random local search (RLS). By [54, Theorem 1], w.h.p. RLS needs time $\Omega(n\ln d')$ to find the optimum, which concludes the proof.

Claim *(c)* follows directly from the proof of Theorem 3.2, using the parameter $\alpha = 2$. There it was shown that with the potential $g(x) = \sum_{i=1}^{n} g_i x_i$, the drift towards the optimum is at least $\frac{p_1}{2n} \cdot g(x)$ by (6). By design, the minimal positive potential is 1. Since we start in distance one from the optimum, the initial potential is at most

$$g_{\text{init}} \leq \max\{g_i : i \in [n]\} = g_n \leq \gamma_n, \tag{13}$$

where

$$\gamma_n = \left(1 + \frac{2\chi^3}{(n-1)p_1^2}\right)^{n-1} \leq \exp\left(\frac{2\chi^3}{p_1^2}\right) = O(1) \tag{14}$$

by (4) and (5). Hence, the expected runtime is at most

$$\mathbb{E}[T] \leq \frac{\ln(g_{\text{init}}) + 1}{p_1/(2n)} = \frac{O(n)}{p_1} = O(n) \tag{15}$$

by the multiplicative drift theorem. □

We did not aim for tightness in Proposition 3.6, but rather want to demonstrate the different regimes. In particular, consider a distribution $\mathcal{D}$ with $p_1 = \Theta(1)$ and with mean χ. If $\chi = O(1)$, then $\mathbb{E}[T^x_{\mathcal{D}}(f)] = O(n)$ by (c), but for $\chi = \omega(1)$ we have $\mathbb{E}[T^x_{\mathcal{D}}(f)] = \omega(n)$ by (a). This shows that the size of χ is truly relevant for the runtime, at least if the algorithms starts in an adversarial point. Moreover, (b) shows that the high expectation in the case $\chi = \omega(1)$ is not just due to low-probability events, but that it comes from typical runs.

The most interesting and common unbiased mutation operators, except for standard-bit mutation, are mutation operators where the number of bit flips has a *heavy tail*. Usually a *power law* is used, i.e, the probability to flip k bits scales like $k^{-\alpha}$ for some constant $\alpha > 1$. There are two different regimes for the parameter α. For $\alpha > 2$, the expected number of bit flips satisfies $\chi = O(1)$. For $\alpha \in (1, 2)$, the expected number of bit flips is unbounded and large, $\chi = n^{\Omega(1)}$.[2] In either case, such power-law distributions satisfy $p_1 = \Theta(1)$. Notably, our main Theorem 1.1 applies to power-law distributions with $\alpha > 2$, but not to power-law distributions with $\alpha \in (1, 2)$. We believe that this reflects a real difference between those two cases. As indication, note that in the situation of Proposition 3.6, we have $\mathbb{E}[T^x_{\mathcal{D}}(f)] = O(n)$ for $\alpha > 2$, but $\mathbb{E}[T^x_{\mathcal{D}}(f)] = \Omega(n \ln n)$ for $\alpha \in (1, 2)$. This does not rule out that Theorem 1.1 still might be true for $\alpha \in (1, 2)$ due to the random starting point, but it suggests that trajectories of the algorithm can be substantially different.

4 LOWER BOUND

The following theorem is the main result shown in this section.

Theorem 4.1. *Let $\mathcal{D} = (p_0, p_1, \dots, p_n)$ be a probability distribution on $[0, n]$ with $p_1 + p_{n-1} = n^{-o(1)}$. Let $f : \{0,1\}^n \to \mathbb{R}$ be a function with a unique global optimum. Then the expected runtime of the $(1+1)$-EA$_{\mathcal{D}}$ on f is bounded from below by*

$$(1 - o(1)) \frac{1}{p_1 + p_{n-1}} n \ln n. \tag{16}$$

Note that most common mutation operators satisfy $p_{n-1} = o(p_1)$, in which case (16) simplifies to $(1 - o(1)) \frac{1}{p_1} n \ln n$. We remark that a coarser lower bound of $\Omega(n \ln n)$ follows from [42] and [20]. However, in contrast to [42], we are interested in understanding the leading constant, and in contrast to [20], we are interested in the expected runtime for *static* unary unbiased distributions. A common technique to prove lower bounds that apply to any function from some problem collection is to bound the expected runtime of the algorithm on OneMax and to show that OneMax is the "easiest" among all functions from the collection, in the sense that the expected runtime of the algorithm optimizing a given function from the set cannot be smaller than its expected runtime on OneMax. In many cases, e.g., when considering the $(1+1)$-EA with standard bit mutation, OneMax can even be shown to be the easiest among all functions with unique global optimum; as was first shown in [25] for mutation rate $p = 1/n$ and then in [53] more generally for all (static or dynamic) mutation rates $p \leq 1/2$. However, in our situation it is not true that OneMax is the easiest function, as we will discuss in Section 4.2.

Our proof for Theorem 4.1 follows the strategy used in [20]. In particular, we apply their lower bound theorem for variable drift [20, Theorem 9] in the same way. We quote their Lemma 13 directly, and the proof of our Theorem 4.8 below differs from the proof of their Theorem 14 only in the calculations and bounds used. The key difference between their proof and ours is that we use a different function h to bound the expected change in the potential. Most of the work goes into showing that this function h is indeed applicable, and providing an upper bound on its values that allows us to translate the result of Theorem 4.8 into the asymptotic formulation of Theorem 4.1. In this sketch, we give the key steps, but omit the proofs of the intermediate lemmas.

To implement the proof strategy of [20], we use the same potential function to measure the progress of the optimization process. That is, we denote by $(x^{(0)}, x^{(1)}, \dots, x^{(t)})$ the sequence of the first $t+1$ search points evaluated by the algorithm and we denote by $v_t \in \{x^{(0)}, \dots, x^{(t)}\}$ the parent chosen by the algorithm in iteration t. We define the potential at time t as

$$X_t := \min_{0 \leq i \leq t} d\big(x^{(i)}\big), \tag{17}$$

where d is the distance function

$$d(x) := \min\{n - \textsc{Om}(x), \textsc{Om}(x)\}. \tag{18}$$

The reason for considering the symmetric distance to the optimum and its complement is that an optimal unary unbiased black-box algorithm may first reach $\vec{0}$, and then flip all bits at once. Furthermore, as we show in Lemma 4.9, it is possible to make progress towards the optimum in a way that can be measured in terms of d, while the Hamming-distance to the optimum increases.

Note that the sequence $(X_t)_{t \geq 0}$ is non-increasing, so we may apply the variable drift lower bound from [20, Theorem 9] to it.

Next, we define the function $\tilde{h}$, which gives the precise drift in the case where the algorithm uses a bitstring at distance X_t for generating the offspring in round t.

Definition 4.2. We define $\tilde{h} : [0, n] \to \mathbb{R}_{\geq 0}$ as

$$\tilde{h}(d) = \sum_{r=1}^{n-1} (p_r + p_{n-r}) B(n, d, r), \tag{19}$$

[2] For $\alpha = 2$ the expected number of bit flips is also unbounded, but grows only as $\chi = O(\ln n)$. We will neglect this case here.

where

$$B(n,d,r) = \sum_{i=\max\{\lceil r/2\rceil, r+d-n\}}^{\min\{d,r\}} (2i-r)\frac{\binom{d}{i}\binom{n-d}{r-i}}{\binom{n}{r}} \tag{20}$$

is the drift conditioned on flipping r bits.

The expression $B(n,d,r)$ was already given in [20] as the exact fitness drift with respect to OneMax when flipping r bits.

Lemma 4.3. *If a static unary unbiased algorithm with flip distribution $\mathcal{D}$ chooses a bitstring v_t with potential X_t for mutation in step t, then the drift is given by $\tilde{h}(X_t)$, i.e.,*

$$\mathbb{E}\left[X_t - X_{t+1} \mid \{X_t = d\} \wedge \{d(v_t) = d\}\right] = \tilde{h}(d). \tag{21}$$

The proof relies on the fact that flipping $n-r$ bits is the same as first flipping n bits and then flipping r bits to adapt the computation of $B(n,d,r)$ given in [20].

Now, we are ready to define the bound h on the drift that we use in our application of the variable drift theorem [20, Theorem 9].

Definition 4.4. Let $h : [0,n] \to \mathbb{R}_{\geq 0}$,

$$h(d) = \begin{cases} \tilde{h}(d), & \text{for } d \leq d_0 \\ n, & \text{for } d > d_0, \end{cases} \tag{22}$$

where

$$d_0 := \lfloor (p_1 + p_{n-1})n/\ln^2 n \rfloor. \tag{23}$$

The following statement is adapted from Lemma 21 in [20].

Lemma 4.5. *There is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, $d \leq d_0$, and $r \geq r_0 = 12$, it holds*

$$B(n,d,r) < (d/n)^2. \tag{24}$$

The next lemma gives an upper bound on $h(d)$ that holds once d is small enough.

Lemma 4.6. *There is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, and all $d \leq d_0$,*

$$h(d) \leq \left(1 + \frac{1}{\ln n}\right) \cdot (p_1 + p_{n-1}) \cdot \frac{d}{n}. \tag{25}$$

As we show in the following lemma, the function h is indeed an upper bound for the change in the potential. We need to show that, under our assumptions, the expected change in the potential conditioning on $d(v_t) = d + \Delta$ is maximal if $\Delta = 0$. The proof relies on a case distinction to deal with different ranges of d and Δ. Depending on the case, we use an additive Chernoff bound, a multiplicative Chernoff bound, or Lemma 4.6.

Lemma 4.7. *There is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, any static unary unbiased algorithm with flip distribution $\mathcal{D}$ such that $p_1 + p_{n-1} = n^{-o(1)}$, and all $d \leq d_0$, it holds*

$$\mathbb{E}\left[X_t - X_{t+1} \mid X_t = d\right] \leq h(d). \tag{26}$$

Finally, we show that there is an $n_0 \in \mathbb{N}$ such that for all $n \geq n_0$, the function h is monotonically increasing. With this statement at hand and Lemma 13 from [20], which bounds the probability to make large jumps, we can then show the following theorem, using very similar computations as those that were used in [20].

Theorem 4.8. *The expected runtime of any static unary unbiased algorithm with flip distribution $\mathcal{D}$ satisfying $p_1 + p_{n-1} = n^{-o(1)}$ on any function $f : \{0,1\}^n \to \mathbb{R}$ with unique global optimum is at least*

$$\sum_{d=1}^{d_0} \frac{1}{h(d)} - o(n). \tag{27}$$

With this statement at hand, we can finally prove Theorem 4.1.

Proof of Theorem 4.1. By Lemma 4.6, we have for all $1 \leq d \leq d_0 = n\frac{p_1+p_{n-1}}{\ln^2 n}$,

$$\frac{1}{h(d)} \geq \frac{n}{(p_1+p_{n-1})d} - \frac{\frac{1}{\ln n}}{1+\frac{1}{\ln n}} \frac{n}{(p_1+p_{n-1})d} \tag{28}$$

$$= (1-o(1))\frac{n}{(p_1+p_{n-1})d}. \tag{29}$$

Applying Theorem 4.8 yields

$$\mathbb{E}[T] \geq \left((1-o(1))\frac{n}{(p_1+p_{n-1})}\sum_{d=1}^{d_0}\frac{1}{d}\right) - o(n) \tag{30}$$

$$= (1 \pm o(1))\frac{1}{(p_1+p_{n-1})} n \ln n. \tag{31}$$

□

4.1 On p_{n-1}

The following lemma shows that the term p_{n-1} in Theorem 4.1 is really necessary. In particular, there are (artificial) functions on which a unary unbiased (1 + 1) algorithm with $p_1 = 0$ can be as efficient as random local search (RLS) on OneMax.

Lemma 4.9. *Let n be even. Let RLS be the $(1+1)$-$EA_{\mathcal{D}}$ with $p_1 = 1$ and $p_i = 0$ for $i \neq 1$, and let $\mathcal{A}$ be the $(1+1)$-$EA_{\mathcal{D}}$ with $p_{n-1} = 1$ and $p_i = 0$ for $i \neq n-1$. Let $f : \{0,1\}^n \to \mathbb{R}$ be defined via*

$$f(x) := \begin{cases} \textsc{Om}(x), & \text{if } \textsc{Om}(x) \text{ is even,} \\ n - \textsc{Om}(x), & \text{if } \textsc{Om}(x) \text{ is odd.} \end{cases}$$

Let $T^{RLS}(\textsc{Om})$ be the runtime of RLS on OneMax, and let $T^{\mathcal{A}}(f)$ be the runtime of $\mathcal{A}$ on f. Then $T^{RLS}(\textsc{Om})$ and $T^{\mathcal{A}}(f)$ follow the same distribution, i.e., for all $T \in \mathbb{N}$,

$$\Pr[T^{RLS}(\textsc{Om}) = T] = \Pr[T^{\mathcal{A}}(f) = T]. \tag{32}$$

In particular, $T^{\mathcal{A}}(f) = (1 \pm o(1))n \ln n$ in expectation and with high probability.

Proof. Let X_t^{RLS} and $X_t^{\mathcal{A}}$ be the fitness of RLS and $\mathcal{A}$ after t iterations, respectively. We will show by induction over t that those two random variables follow the same distribution.

Note that f is obtained from OneMax by swapping the fitness levels k and $n-k$ if k is odd. In particular, the number of search points of fitness k does not change. Since the initial search point is chosen uniformly at random, therefore X_0^{RLS} and $X_0^{\mathcal{A}}$ follow the same distribution.

Now let $t \geq 0$ and assume that X_t^{RLS} and $X_t^{\mathcal{A}}$ follow the same distribution. We claim that for every $k, k' \in [0..n]$, we have

$$\Pr[X_{t+1}^{\mathrm{RLS}} = k' \mid X_t^{\mathrm{RLS}} = k] = \Pr[X_{t+1}^{\mathcal{A}} = k' \mid X_t^{\mathcal{A}} = k]. \tag{33}$$

Note that this implies that X^{RLS}_{t+1} and $X^{\mathcal{A}}_{t+1}$ follow the same distribution since then

$$\begin{aligned}\Pr[X^{\mathrm{RLS}}_{t+1} = k'] &= \sum_{k\in[0,n]} \Pr[X^{\mathrm{RLS}}_t = k] \cdot \Pr[X^{\mathrm{RLS}}_{t+1} = k' \mid X^{\mathrm{RLS}}_t = k] \\ &= \sum_{k\in[0,n]} \Pr[X^{\mathcal{A}}_t = k] \cdot \Pr[X^{\mathcal{A}}_{t+1} = k' \mid X^{\mathcal{A}}_t = k] \\ &= \Pr[X^{\mathcal{A}}_{t+1} = k'].\end{aligned}$$

Moreover, since this implies that X^{RLS}_t and $X^{\mathcal{A}}_t$ follow the same distribution for all t, by

$$\begin{aligned}\Pr[T^{\mathrm{RLS}}(\textsc{Om}) > T] &= \Pr[X^{\mathrm{RLS}}_T < n] \\ &= \Pr[X^{\mathcal{A}}_T < n] = \Pr[T^{\mathcal{A}}(f) > T],\end{aligned}$$

it also implies the lemma. So it remains to show (33).

For RLS it is obvious that the left hand side of (33) is zero for all $k' \in [0..n] \setminus \{k, k+1\}$. Let us assume that $\mathcal{A}$ is in a search point x such that $X^{\mathcal{A}}_t = k$. The algorithm $\mathcal{A}$ creates offspring y by randomly flipping $n-1$ positions of x. This can be equivalently expressed by first flipping all n positions, and then flipping back a uniformly random position. Flipping all n positions yields the antipodal search point x' with $\textsc{Om}(x') = n - \textsc{Om}(x)$. Since y is obtained from x' by flipping exactly one bit, it satisfies $\textsc{Om}(y) = n - \textsc{Om}(x) \pm 1$. Since n is even, this implies that $\textsc{Om}(x')$ and $\textsc{Om}(x)$ are either both odd or both even. In either case, $f(x') = n - f(x)$ and thus $f(y) = f(x) \pm 1$ by definition of f. Since $\mathcal{A}$ is elitist, it will reject any offspring of fitness $f(x) - 1$, so it accepts y if and only if $f(y) = f(x) + 1$. In particular, this means that the right hand side of (33) is zero for all $k' \in [0..n] \setminus \{k, k+1\}$, as required.

For the remaining values $k' \in \{k, k+1\}$, it suffices to show equality for one of them, since the left and right hand side of (33) both sum up to one if summed over all k'. If $\textsc{Om}(x)$ is even, then the offspring is fitter if and only if the bit that is *not* flipped is a zero-bit, which happens with probability $(n - \textsc{Om}(x))/n = (n-k)/n$. If $\textsc{Om}(x)$ is odd, then the offspring is fitter if and only if the bit that is not flipped is a one-bit, which happens with probability $\textsc{Om}(x)/n = (n-k)/n$. So in either case, $\Pr[X^{\mathcal{A}}_{t+1} = k+1 \mid X^{\mathcal{A}}_t = k] = (n-k)/n$, which is the same as the probability for RLS. This concludes the proof of (33) and of the lemma. □

The following theorem strengthens Theorem 4.1 for the $(1+1)$-$\mathrm{EA}_{\mathcal{D}}$ on linear functions. It says that in this case, p_{n-1} does not help to improve the asymptotic runtime.

Theorem 4.10. *Consider the $(1+1)$-$EA_{\mathcal{D}}$ with distribution $\mathcal{D} = (p_0, p_1, \dots, p_n)$ such that $p_1 = n^{-o(1)}$. The expected runtime on any linear function on $\{0,1\}^n$ is at least*

$$(1 - o(1))\frac{1}{p_1} n \ln n. \tag{34}$$

Proof. Recall that the weights w_i of f are positive and sorted. We may assume that the smallest weight is $w_1 = 1$, since we can multiply all weights with the same constant factor without changing the fitness landscape. Moreover, for linear functions it is slightly more convenient to work with minimization instead of maximization. Both versions are equivalent, so we may assume that f is minimized. Finally, if $w_n > \sum_{i=1}^{n-1} w_i + 1$ then replacing w_n by $\sum_{i=1}^{n-1} w_i + 1$ does not change the fitness landscape since in either case all search points x with $x_n = 1$ have higher objective than all search points with $x_n = 0$. Hence we may assume $w_n \le \sum_{i=1}^{n-1} w_i + 1$. Writing $W := \sum_{i=1}^{n} w_i$ for the total weight, this implies $2w_n \le W + 1 < \frac{3}{2}W$, and thus $w_n < \frac{3}{4}W$.

Fix some t, and let $q_{i,t} := \Pr[x_i^{(t)} = 1]$ be the probability that the i-th bit is a one-bit in generation t. Then a classical result by Jägersküpper [39] says that $q_{1,t} \ge \dots \ge q_{n,t}$. Jägersküpper proved it for the $(1+1)$-EA with standard bit mutation, but the only ingredient in the proof was that for all $i, j \in [n]$, if we condition on the set of flips in $[n] \setminus \{i, j\}$ then positions i and j have the same probability of being flipped. This is true for all unbiased mutation operators, so Jägersküpper's result holds for the $(1+1)$-$\mathrm{EA}_{\mathcal{D}}$ as well. Moreover, the proof shows inductively for all times that for any substring $\tilde{x}$ on the positions $[n] \setminus \{i, j\}$, the combination "$x_i = 0$, $x_j = 1$, $\tilde{x}$" is more likely than the combination "$x_i = 1$, $x_j = 0$, $\tilde{x}$" if $i < j$. Since both options have the same number of one-bits, and since other options (with $x_i = x_j$) contribute equally to $q_{i,t}$ and $q_{j,t}$, it was already observed in [44] that $q_{i,t} \ge q_{j,t}$ still holds if we condition on the number of one-bits $\textsc{Om}(x^{(t)})$ at time t. Moreover, the statement also still holds if we replace t by the hitting time $T = T(d) = \min\{t \ge 0 \mid \textsc{Om}(x^t) \le d\}$, so we have $q_{1,T} \ge \dots \ge q_{n,T}$.

We choose $T = T(d)$ for $d = n/\ln n$. Then we have $\sum_{i\in[n]} q_{i,T} = \mathbb{E}[\textsc{Om}(x^T)] \le d$ by definition of T, and hence

$$\begin{aligned}\mathbb{E}[f(x^{(T)})] = \sum_{i\in[n]} w_i \cdot q_{i,T} &\le \frac{\left(\sum_{i\in[n]} w_i\right) \cdot \left(\sum_{i\in[n]} q_{i,T}\right)}{n} \\ &\le \frac{Wd}{n} = \frac{W}{\ln n},\end{aligned}$$

where the second step is Chebyshev's sum inequality, since w_i and $q_{i,T}$ are sorted opposingly.

By Markov's inequality, at time T we have w.h.p. $f(x^{(T)}) \le W/8$. In the following we will condition on this event. We claim that then after time T, any offspring obtained by an $(n-1)$-bit flip is rejected. To see this, consider any x with $f(x) \le W/8$. Any offspring y that is obtained from x by an $(n-1)$-bit flip has objective $f(y) \ge W - W/8 - w_n$, because the antipodal point of x has objective $W - f(x) \ge W - W/8$, and flipping back a bit can decrease the objective by at most $w_n < \frac{3}{4}W$. Hence, $f(y) \ge W - W/8 - w_n > W/8$. Therefore, the offspring y has higher (worse) objective, and is rejected. Hence, once the algorithm reaches objective at most $W/8$, all offspring obtained from $(n-1)$-bit flips are rejected. In other words mutations of $n-1$ bits are idle steps. This means that after time T, the $(1+1)$-$\mathrm{EA}_{\mathcal{D}}$ behaves as the $(1+1)$-$\mathrm{EA}_{\mathcal{D}'}$, where we define $\mathcal{D}' = (p'_0, \dots, p'_n)$ by

$$p'_i := \begin{cases} 0 & \text{if } i = n-1, \\ p_0 + p_{n-1} & \text{if } i = 0, \\ p_i & \text{otherwise.} \end{cases}$$

At time T w.h.p. we have $\textsc{Om}(x^{(T)}) \ge d - \ln^2 n$, which follows from [20, Lemma 13]. By Theorem 4.1, the $(1+1)$-$\mathrm{EA}_{\mathcal{D}'}$ needs in expectation at least $(1 - o(1))\frac{1}{p'_1 + p'_{n-1}} n \ln n = (1 - o(1))\frac{1}{p_1} n \ln n$ steps to find the optimum from level $d - \ln^2 n$, and hence the $(1+1)$-$\mathrm{EA}_{\mathcal{D}}$ needs the same time. Note that we proved the lower bound

conditional on w.h.p.events, but this just adds another $(1 - o(1))$ factor for the unconditional expectation. □

4.2 No Stochastic Domination

Earlier work [24, 52, 53] used stochastic domination arguments (cf. [15]) to prove lower bounds. In particular, Witt proved his lower bound by showing that OneMax is the easiest function for the $(1 + 1)$-EA with standard bit mutation of arbitrary mutation rate $p \leq 1/2$ [53]. The key ingredient was Lemma 6.1 in [53], which considered offspring y and y' that are created from x and x' respectively by standard bit mutation with mutation rate $p \leq 1/2$. For minimization, if $\textsc{Om}(x) \leq \textsc{Om}(x')$ then the lemma states $\Pr[\textsc{Om}(y) \leq k] \geq \Pr[\textsc{Om}(y') \leq k]$ for all $k \in [0, n]$. So it is easier to reach Om-level at most k when starting with a parent of smaller Om-value. This lemma implies on the one hand that OneMax is the easiest function for standard bit mutation, but also that elitist selection is optimal in this situation: the $(1 + 1)$-EA with mutation rate $p \leq 1/2$ is the fastest algorithm on OneMax among all unary algorithms using standard bit mutation with mutation rate $p \leq 1/2$.

However, Witt's lemma does not hold for general unbiased mutation operators. In particular, being closer to the optimum does not mean that we have a higher chance of finding the optimum in the next step. Consider the case where the algorithm flips one bit with probability $p_1 = n^{-2}$ and two bits with probability $p_2 = 1 - p_1 = 1 - n^{-2}$. The probability of finding the optimum from a search point in Hamming distance one from the optimum is $p_1/n = n^{-3}$, whereas the probability of finding the optimum from a search point in Hamming distance two from the optimum is $p_2/\binom{n}{2} = \Theta(n^{-2})$, which is much larger.

Even worse, let us consider the time T_d to find the optimum on OneMax if we start in Hamming distance d. For $d = 1$ we have $\mathbb{E}[T_1] = n/p_1 = \Theta(n^3)$. For $d = 2$, the probability of making an improvement is $p_{\text{imp}} = p_1 \cdot 2/n + p_2/\binom{n}{2} = \Theta(n^{-2})$. Hence, conditional on making an improvement, the algorithm improves by one with probability $(p_1 \cdot 2/n)/p_{\text{imp}} = \Theta(n^{-1})$. Therefore, we need to wait in expectation $1/p_{\text{imp}} = \Theta(n^2)$ rounds for an improvement, and with probability $\Theta(n^{-1})$ we improve only by one and need to wait another T_1 rounds for reaching the optimum. Hence,

$$\mathbb{E}[T_2] = \Theta(n^2) + \Theta(n^{-1}) \cdot \mathbb{E}[T_1] = \Theta(n^2),$$

which is asymptotically smaller than $\mathbb{E}[T_1] = \Theta(n^3)$. So the expected time $\mathbb{E}[T_d]$ is not monotone in d, and can be asymptotically smaller if we start further away from the optimum.

Turning this example around, we can construct a situation where OneMax is not the easiest function. Consider an algorithm with $p_1 = n^{-3}$, $p_2 = n^{-1}$ and $p_3 = 1 - p_1 - p_2 = 1 - o(1)$, starting in the string $x = (01 \dots 1)$ where all but the first bit are optimized. On OneMax, it needs to wait for a one-bit flip, which takes time $n/p_1 = \Theta(n^4)$. But if the fitness function is $f(x) := 3x_1 + \sum_{i=2}^{n} x_i$, then the algorithm accepts any mutation flipping two or three bits if it involves x_1. Conditional on flipping x_1, a two-bit flip has only probability $O(n^{-1})$ since $p_3/p_2 = \Theta(n)$. In that case (an improving two-bit flip) the algorithm jumps to another neighbour of the optimum and needs to wait $n/p_1 = O(n^4)$ rounds for the right one-bit flip. This contributes $O(n^{-1} \cdot n^4) = O(n^3)$ to the expectation. However, in the more likely case of a three-bit flip, the algorithm jumps to a search point in distance two from the optimum. By a similar calculation as before, it now needs time $O(n^3)$ to find the optimum, so the expected runtime on f is $O(n^3)$, which is asymptotically faster than on OneMax.

Finally, the same example can be used to show that a non-elitist $(1 + 1)$ algorithm may be faster than the $(1 + 1)$-EA$_{\mathcal{D}}$ if both use the same unbiased mutation operator. Hence, Witt's lemma and all its consequences fail for general unbiased mutation operators. This is similar to the situation for the compact genetic algorithm cGA, for which this form of domination also does not hold [16].

5 CONCLUSIONS

We have extended Witt's result bounding the runtime of the $(1+1)$-EA on linear functions to arbitrary elitist (1+1) unary unbiased EAs and we have discussed various ways in which the requirements made in Corollary 3.3 and Theorem 4.1 are tight. In particular, we have seen that for $p_1 = n^{-\Omega(1)}$, the expected runtime can be smaller than $\frac{1}{p_1} n \ln n$ by a constant factor. When interpreted in the light of black-box complexity, our results can be seen as extensions of [20] to linear functions. However, we have focused in this work on *static* mutation operators. An extension of our result to *dynamic* parameter settings would hence be a natural continuation of our work.

Another direction in which we aim to extend our results are *combinatorial* optimization problems where we suspect to see a tangible advantage of unusual unary mutation operators. For example, the optimal mutation operator for the minimum spanning tree problem (MST) is likely to satisfy $p_1 > 0$ and $p_2 > 0$. Similarly, there are functions like LeadingOnes where the optimal number of flipped bits depends on the phase of the algorithm, and none of the phases is asymptotically negligible for the runtime. In such cases, it may be interesting to see what the optimal distribution is.

Similarly, we also expect advantages of the $(1 + 1)$-EA$_{\mathcal{D}}$ over standard $(1 + 1)$-EAs when optimizing for average performance for problem collections with instances having different landscapes.

ACKNOWLEDGMENTS

Our work is financially supported by ANR-22-ERCS-0003-01 project VARIATION. Duri Andrea Janett was supported by the Swiss-European Mobility Programme.

REFERENCES

[1] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. 2021. Lazy parameter tuning and control: choosing all parameters randomly from a power-law distribution. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1115–1123.

[2] Denis Antipov, Maxim Buzdalov, and Benjamin Doerr. 2022. Fast mutation in crossover-based algorithms. *Algorithmica* 84 (2022), 1724–1761.

[3] Denis Antipov and Benjamin Doerr. 2020. Runtime analysis of a heavy-tailed $(1 + (\lambda, \lambda))$ genetic algorithm on jump functions. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II*. Springer, 545–559.

[4] Denis Antipov and Benjamin Doerr. 2021. Precise runtime analysis for plateau functions. *ACM Transactions on Evolutionary Learning and Optimization* 1, 4 (2021), 1–28.

[5] Denis Antipov and Semen Naumov. 2021. The effect of non-symmetric fitness: The analysis of crossover-based algorithms on RealJump functions. In *Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. 1–15.

[6] Henry Bambury, Antoine Bultel, and Benjamin Doerr. 2021. Generalized jump functions. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 1124–1132.

[7] Henry Bambury, Antoine Bultel, and Benjamin Doerr. 2022. An Extended Jump Functions Benchmark for the Analysis of Randomized Search Heuristics. *Algorithmica* (2022), 1–32.

[8] Pauline Bennet, Carola Doerr, Antoine Moreau, Jeremy Rapin, Fabien Teytaud, and Olivier Teytaud. 2021. Nevergrad: black-box optimization platform. *ACM SIGEVOlution* 14, 1 (2021), 8–15.

[9] Nathan Buskulic and Carola Doerr. 2021. Maximizing Drift Is Not Optimal for Solving OneMax. *Evol. Comput.* 29, 4 (2021), 521–541.

[10] Maxim Buzdalov. 2022. The (1+(lambda,lambda)) Genetic Algorithm on the Vertex Cover Problem: Crossover Helps Leaving Plateaus. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–10.

[11] Maxim Buzdalov and Carola Doerr. 2020. Optimal Mutation Rates for the (1+λ) EA on OneMax. In *Proc. of Parallel Problem Solving from Nature (PPSN) (Lecture Notes in Computer Science, Vol. 12270)*. Springer, 574–587.

[12] Maxim Buzdalov and Carola Doerr. 2021. Optimal static mutation strength distributions for the (1+ λ) evolutionary algorithm on OneMax. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 660–668.

[13] Dogan Corus, Pietro S Oliveto, and Donya Yazdani. 2021. Automatic adaptation of hypermutation rates for multimodal optimisation. In *Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms*. 1–12.

[14] Dogan Corus, Pietro S Oliveto, and Donya Yazdani. 2021. Fast immune system-inspired hypermutation operators for combinatorial optimization. *IEEE Transactions on Evolutionary Computation* 25, 5 (2021), 956–970.

[15] Benjamin Doerr. 2019. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science* 773 (2019), 115–137.

[16] Benjamin Doerr. 2021. The Runtime of the Compact Genetic Algorithm on Jump Functions. *Algorithmica* 83 (10 2021), 1–49.

[17] Benjamin Doerr and Carola Doerr. 2014. Reducing the arity in unbiased black-box complexity. *Theoretical Computer Science* 545 (2014), 108–121.

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

[19] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2015. Unbiased black-box complexities of jump functions. *Evolutionary Computation* 23, 4 (2015), 641–670.

[20] Benjamin Doerr, Carola Doerr, and Jing Yang. 2020. Optimal parameter choices via precise black-box analysis. *Theoretical Computer Science* 801 (2020), 1–34.

[21] Benjamin Doerr, Yassine Ghannane, and Marouane Ibn Brahim. 2022. Runtime Analysis for Permutation-based Evolutionary Algorithms. *arXiv preprint arXiv:2207.04045* (2022).

[22] Benjamin Doerr, Omar El Hadri, and Adrien Pinard. 2022. The (1+(λ, λ)) global SEMO algorithm. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 520–528.

[23] Benjamin Doerr, Daniel Johannsen, Timo Kötzing, Per Kristian Lehre, Markus Wagner, and Carola Winzen. 2011. Faster black-box algorithms through higher arity operators. In *Proceedings of the 11th workshop proceedings on Foundations of genetic algorithms*. 163–172.

[24] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2010. Drift analysis and linear functions revisited. In *IEEE Congress on Evolutionary Computation*. 1–8.

[25] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative Drift Analysis. *Algorithmica* 64 (2012), 673–697.

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

[27] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast genetic algorithms. In *Genetic and Evolutionary Computation Conference (GECCO)*.

[28] Benjamin Doerr and Frank Neumann. 2021. A Survey on Recent Progress in the Theory of Evolutionary Algorithms for Discrete Optimization. *ACM Trans. Evol. Learn. Optim.* 1, 4 (2021), 16:1–16:43.

[29] Benjamin Doerr and Zhongdi Qu. 2022. A first runtime analysis of the NSGA-II on a multimodal problem. In *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part II*. Springer, 399–412.

[30] Benjamin Doerr and Amirhossein Rajabi. 2022. Stagnation detection meets fast mutation. *Theoretical Computer Science* (2022).

[31] Benjamin Doerr and Carola Winzen. 2012. Black-box complexity: Breaking the O (n log n) barrier of LeadingOnes. In *Artificial Evolution: 10th International Conference, Evolution Artificielle, EA 2011, Angers, France, October 24-26, 2011, Revised Selected Papers 10*. Springer, 205–216.

[32] Benjamin Doerr and Weijie Zheng. 2021. Theoretical Analyses of Multi-Objective Evolutionary Algorithms on Multi-Modal Objectives. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 35. 12293–12301.

[33] Carola Doerr. 2020. Complexity theory for discrete black-box optimization heuristics. *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization* (2020), 133–212.

[34] Carola Doerr, Duri Andrea Janett, and Johannes Lengler. 2023. Tight Runtime Bounds for Static Unary Unbiased Evolutionary Algorithms on Linear Functions. *CoRR* abs/2302.12338 (2023). https://doi.org/10.48550/arXiv.2302.12338 arXiv:2302.12338

[35] Carola Doerr, Hao Wang, Furong Ye, Sander Van Rijn, and Thomas Bäck. 2018. IOHprofiler: A benchmarking and profiling tool for iterative optimization heuristics. *arXiv preprint arXiv:1810.05281* (2018).

[36] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2006. Upper and Lower Bounds for Randomized Search Heuristics in Black-box Optimization. *Theory of Computing Systems* 39 (2006), 525–544.

[37] Tobias Friedrich, Andreas Göbel, Francesco Quinzan, and Markus Wagner. 2018. Heavy-tailed mutation operators in single-objective combinatorial optimization. In *Parallel Problem Solving from Nature–PPSN XV: 15th International Conference, Coimbra, Portugal, September 8–12, 2018, Proceedings, Part I 15*. Springer, 134–145.

[38] Tobias Friedrich, Francesco Quinzan, and Markus Wagner. 2018. Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 293–300.

[39] Jens Jägersküpper. 2008. A blend of Markov-chain and drift analysis. In *Parallel Problem Solving from Nature–PPSN X: 10th International Conference, Dortmund, Germany, September 13-17, 2008. Proceedings 10*. Springer, 41–51.

[40] Jaroslav Klapálek, Antonín Novák, Michal Sojka, and Zdeněk Hanzálek. 2021. Car Racing Line Optimization with Genetic Algorithm using Approximate Homeomorphism. In *2021 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. IEEE, 601–607.

[41] Per Kristian Lehre and Dirk Sudholt. 2019. Parallel black-box complexity with tail bounds. *IEEE Transactions on Evolutionary Computation* 24, 6 (2019), 1010–1024.

[42] Per Kristian Lehre and Carsten Witt. 2012. Black-Box Search by Unbiased Variation. *Algorithmica* 64 (2012), 623–642.

[43] Johannes Lengler. 2019. A general dichotomy of evolutionary algorithms on monotone functions. *IEEE Transactions on Evolutionary Computation* 24, 6 (2019), 995–1009.

[44] Johannes Lengler and Nicholas Spooner. 2015. Fixed budget performance of the (1+ 1) EA on linear functions. In *Proceedings of the 2015 ACM Conference on Foundations of Genetic Algorithms XIII*. 52–61.

[45] Vladimir Mironovich and Maxim Buzdalov. 2017. Evaluation of heavy-tailed mutation operator on maximum flow test generation problem. In *Proceedings of the Genetic and Evolutionary Computation Conference Companion*. 1423–1426.

[46] Aneta Neumann, Denis Antipov, and Frank Neumann. 2022. Coevolutionary Pareto diversity optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference*. 832–839.

[47] Aneta Neumann, Yue Xie, and Frank Neumann. 2022. Evolutionary algorithms for limiting the effect of uncertainty for the knapsack problem with stochastic profits. In *Parallel Problem Solving from Nature–PPSN XVII: 17th International Conference, PPSN 2022, Dortmund, Germany, September 10–14, 2022, Proceedings, Part I*. Springer, 294–307.

[48] Antonin Novak, Premysl Sucha, Matej Novotny, Richard Stec, and Zdenek Hanzalek. 2022. Scheduling jobs with normally distributed processing times on parallel machines. *European Journal of Operational Research* 297, 2 (2022), 422–441.

[49] Artem Pavlenko, Daniil Chivilikhin, and Alexander Semenov. 2022. Asynchronous Evolutionary Algorithm for Finding Backdoors in Boolean Satisfiability. In *2022 IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 1–8.

[50] Francesco Quinzan, Andreas Göbel, Markus Wagner, and Tobias Friedrich. 2021. Evolutionary algorithms and submodular functions: benefits of heavy-tailed mutations. *Natural Computing* (2021), 1–15.

[51] Alexander Semenov, Daniil Chivilikhin, Artem Pavlenko, Ilya Otpuschennikov, Vladimir Ulyantsev, and Alexey Ignatiev. 2021. Evaluating the hardness of SAT instances using evolutionary optimization algorithms. In *27th International Conference on Principles and Practice of Constraint Programming (CP 2021)*. Schloss Dagstuhl-Leibniz-Zentrum für Informatik.

[52] Dirk Sudholt. 2013. A New Method for Lower Bounds on the Running Time of Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 17, 3 (2013), 418–435.

[53] Carsten Witt. 2013. Tight bounds on the optimization time of a randomized search heuristic on linear functions. *Combinatorics, Probability and Computing* 22, 2 (2013), 294–318.

[54] Carsten Witt. 2014. Fitness levels with tail bounds for the analysis of randomized search heuristics. *Inform. Process. Lett.* 114, 1-2 (2014), 38–41.

[55] Mengxi Wu, Chao Qian, and Ke Tang. 2018. Dynamic mutation based Pareto optimization for subset selection. In *Intelligent Computing Methodologies: 14th International Conference, ICIC 2018, Wuhan, China, August 15-18, 2018, Proceedings, Part III 14*. Springer, 25–35.

[56] Furong Ye, Carola Doerr, and Thomas Bäck. 2019. Interpolating Local and Global Search by Controlling the Variance of Standard Bit Mutation. In *Proc. of IEEE Congress on Evolutionary Computation (CEC)*. IEEE, 2292–2299.

[57] Furong Ye, Hao Wang, Carola Doerr, and Thomas Bäck. 2020. Benchmarking a genetic algorithm with configurable crossover probability. In *Parallel Problem Solving from Nature–PPSN XVI: 16th International Conference, PPSN 2020, Leiden, The Netherlands, September 5-9, 2020, Proceedings, Part II*. Springer, 699–713.