Mutation Rate Matters Even When Optimizing Monotonic Functions

Benjamin Doerr
Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

Thomas Jansen t.jansen@cs.ucc.ie
University College Cork, Cork, Ireland

Dirk Sudholt
University of Birmingham, Birmingham B15 2TT, UK

Carola Winzen
Max-Planck-Institut für Informatik, 66123 Saarbrücken, Germany

Christine Zarges zarges@dcs.warwick.ac.uk
University of Warwick, Coventry CV4 7AL, UK

Abstract

Extending previous analyses on function classes like linear functions, we analyze how the simple (1+1) evolutionary algorithm optimizes pseudo-Boolean functions that are strictly monotonic. These functions have the property that whenever only 0-bits are changed to 1, then the objective value strictly increases. Contrary to what one would expect, not all of these functions are easy to optimize. The choice of the constant c in the mutation probability $p(n) = c/n$ can make a decisive difference.

We show that if $c < 1$, then the (1+1) EA finds the optimum of every such function in $\Theta(n \log n)$ iterations. For $c = 1$, we can still prove an upper bound of $O(n^{3/2})$. However, for $c \geq 16$, we present a strictly monotonic function such that the (1+1) EA with overwhelming probability needs $2^{\Omega(n)}$ iterations to find the optimum. This is the first time that we observe that a constant factor change of the mutation probability changes the runtime by more than a constant factor.

Keywords

Evolutionary algorithms, pseudo-Boolean optimization, runtime analysis, computational complexity.

1 Introduction

Rigorously understanding how randomized search heuristics solve optimization problems and proving guarantees for their performance remains a challenging task. The current state of the art is that we can analyze some heuristics for simple problems. Nevertheless, the current words yielded new insight, helped to discount mistaken beliefs, and turned correct beliefs into proven facts.

For example, it was long believed that a pseudo-Boolean function $f : \{0,1\}^n \to \mathbb{R}$ is easy to optimize if it is unimodal, that is, if each $x \in \{0,1\}^n$ that is not optimal has

A preliminary version of this work was presented at the conference *Parallel Problem Solving from Nature (PPSN)* in 2010 (Doerr, Jansen, Sudholt, Winzen, and Zarges, 2010).

a Hamming neighbor y with $f(y) > f(x)$ (Mühlenbein, 1992). Recall that y is called a Hamming neighbor of x if x and y differ in exactly one bit.

This belief was debunked in Droste, Jansen, and Wegener (1998). There the unimodal long k-path function (Horn, Goldberg, and Deb, 1994) was considered and it was proven that the simple (1+1) evolutionary algorithm ((1+1) EA) with high probability does not find the optimum within $2^{\sqrt{n}}$ iterations. Note that, as was seemingly overlooked for a long time, the *Annals of Probability* paper by Aldous (1983) also implies that unimodal functions are not necessarily easy for randomized algorithms. This classical episode shows how important it is to support an intuitive understanding of evolutionary algorithms with rigorous proofs.

It also shows that it is very difficult to identify problem classes that are easy for a particular randomized search heuristic. This, however, is needed for a successful application of such methods, because the no free lunch theorems (Igel and Toussaint, 2004) tell us, in simple words, that no randomized search heuristic can be superior to another if we do not restrict the problem class we are interested in.

1.1 Previous Work

In the following, we restrict ourselves to classes of pseudo-Boolean functions. We stress that the last 10 years also produced a number of results on combinatorial problems, (cf. Oliveto, He, and Yao, 2007). At the same time, research on classical test functions and function classes continued, spurred by the many still open problems.

We also restrict ourselves to one of the most simple randomized search heuristics, the (1+1) EA. The first rigorous results on this heuristic were given by Mühlenbein (1992), who determined how long it takes to find the optimum of simple test functions like $\text{OneMax}(x) := \sum_{i=1}^{n} x_i$, counting the number of 1-bits. Quite some time later, and with much more technical effort necessary, Droste, Jansen, and Wegener (2002) extended the $O(n \log n)$ bound to all linear functions $f(x) := \sum_{i=1}^{n} a_i x_i$. Without loss of generality, one can assume that all coefficients a_i are positive, so that the all 1s string 1^n is the global optimum. Since it was hard to believe that such a simple result should have such a complicated proof, this work initiated a sequence of follow-up results in particular introducing the powerful drift analysis method to the community (He and Yao, 2001, 2002) (see, e.g., Jägersküpper, 2011; Doerr, Johannsen, and Winzen, 2010a; Doerr and Goldberg, 2010b, for recent extensions). However, not all promising-looking function classes are easy to optimize. As laid out in the first paragraphs of this paper, unimodal functions are already difficult.

Almost all of the results described above were proven for the standard mutation probability $1/n$. It is easy to see from their proofs (or, in the case of linear functions, the more elaborate methods needed, Doerr and Goldberg, 2010a), that all results remain true for $p(n) = c/n$, where c can be an arbitrary constant.

We should add that the question of how to determine the right mutation probability is also far from being settled. Most theory results for simplicity take the value $p(n) = 1/n$, but it is known that this is not always optimal (Jansen and Wegener, 2000). In practical applications, $1/n$ is the most recommended static choice for the mutation probability (Bäck, Fogel, and Michalewicz, 1997; Ochoa, 2002) in spite of known limitations of this choice (Cervantes and Stephens, 2009).

Only recently, more precise theoretical results on optimal mutation probabilities have appeared. Böttcher, Doerr, and Neumann (2010) showed that $p(n) \approx 1.59/n$ is the best choice of the mutation probability for the LeadingOnes problem. Witt (2011) proved that $p(n) = 1/n$ is an optimal mutation probability for the (1+1) EA on linear functions. Sudholt (2011) proved the same for the (1+1) EA on long k-paths.

With the standard mutation probability $1/n$ the (1+1) EA is similar to a random local search where exactly one bit is flipped. The bit to be flipped is chosen uniformly at random. The resulting search point replaces the current one in case its fitness is not worse. For many functions, the (1+1) EA with mutation probability $1/n$ and random local search have equal performance. The general question where random local search and the (1+1) EA have asymptotically equal performance is difficult to answer (Doerr, Jansen, and Klein, 2008). But for the class of linear functions, both algorithms have expected optimization time $\Theta(n \log n)$. For random local search, this also holds for monotonic functions for the very same reasons. Flipping a single bit from 0 to 1 implies that the fitness strictly increases and so this 1 will never be lost again. On average, after $O(n \log n)$ mutations, each bit has been flipped at least once, so that each bit has a value of 1. For the (1+1) EA, however, things are much more involved and become very different if the mutation probability is only slightly increased to c/n (with a sufficiently large constant c).

1.2 Our Work

In this work, we regard the class of strictly monotonic functions. A pseudo-Boolean function is called strictly monotonic (or simply monotonic in the following) if any mutation flipping at least one 0-bit into a 1-bit and no 1-bit into a 0-bit strictly increases the function value. Hence, much stronger than for unimodal functions, we not only require that each nonoptimal x has a Hamming neighbor with better f-value, but we even ask that this holds for all Hamming neighbors that have an additional 1-bit.

Obviously, the class of monotonic functions includes linear functions with all bit weights positive. On the other hand, each monotonic function is unimodal. Contrary to the long k-path function, there is always a short path of at most n search points with increasing f-value connecting a search point to the optimum.

It is easy to see that monotonic functions are just the ones where a simple coupon collector argument shows that random local search finds the optimum in time $O(n \log n)$. Surprisingly, we find that monotonic functions are not easy to optimize for the (1+1) EA in general. Secondly, our results show that for this class of functions, the mutation probability $p(n) = c/n$, where c is a constant, can make a crucial difference.

More precisely, we show that for $c < 1$, the (1+1) EA with mutation probability c/n finds the optimum of any monotonic function in time $\Theta(n \log n)$, which is the best possible given previous results on linear functions. For $c = 1$, the drift argument breaks down and we have to resort to an upper bound of $O(n^{3/2})$ based on a related model by Jansen (2007). We currently do not know the full truth. As the lower bound, we only have the general lower bound $\Omega(n \log n)$ for all mutation-based evolutionary algorithms (Droste et al., 2002).

If c is sufficiently large, an unexpected change of regime happens. For $c \geq 16$, we show that there are monotonic functions such that the (1+1) EA with overwhelming probability needs an exponential time to find the optimum. The construction of such functions heavily uses probabilistic methods. To the best of our knowledge, this is the first time that problem instances are constructed this way in the theory of evolutionary computation.

It must be stressed that this is the first result where the mutation probability stays within $\Theta(1/n)$ while the expected optimization time changes from polynomial to exponential. Earlier results showed a similar drastic change only when the order of growth of the mutation probability changed, for example, from $\Theta(\ln(n)/n)$ to $\Theta(1/n)$ (Jansen and Wegener, 2000). In addition, we show that this unexpected behavior may already take place in the class of monotonic functions, which is generally considered to be good natured. From a theoretical point of view, this is an important step toward much more precise results and a better understanding of mutation.

For some randomized search heuristics, our results are also of practical relevance. In memetic algorithms (Krasnogor and Smith, 2005), it is common practice to (occasionally) use very high mutation probabilities to escape regions of local optima. In artificial immune systems (Dasgupta and Niño, 2008) hypermutations using very high mutation probabilities are the most common variation operators. For these algorithms, it is not uncommon to have mutation probabilities that exceed $16/n$. Our results demonstrate the possible danger of this approach. It points out that in general it may not be a good idea to only rely on search by such disruptive variation operators. For artificial immune systems, this has already been pointed out based on theoretical findings (Jansen and Zarges, 2011). That these problems are real has led to the inclusion of less disruptive mutation operators in artificial immune systems in practical applications. For example, in the B-cell algorithm (Kelsey and Timmis, 2003) hypermutations are combined with standard bit mutations using mutation probability $1/n$ leading to proven good performance (Jansen, Oliveto, and Zarges, 2011).

2 Preliminaries

We consider the maximization of a pseudo-Boolean function $f : \{0, 1\}^n \to \mathbb{R}$ by means of a simple evolutionary algorithm, the (1+1) EA. The results can easily be adapted for minimization. In this work, n always denotes the number of bits in the representation. We use common asymptotic notation (Cormen, Leiserson, Rivest, and Stein, 2001); all asymptotics are with respect to n.

The (1+1) EA (see Algorithm 1) maintains a population of size 1. In each generation it creates a single offspring by independently flipping each bit in the current search point with a fixed mutation probability $p(n)$. The new search point replaces the old one in case its f-value is not worse.

Algorithm 1 (1+1) EA with mutation probability $p(n)$

```
1: Initialization: Choose $x \in \{0,1\}^n$ uniformly at random.
2: repeat forever
3:     Create $y \in \{0,1\}^n$ by copying $x$.
4:     Mutation: Flip each bit in $y$ independently with probability $p(n)$.
5:     Selection: if $f(y) \geq f(x)$ then $x := y$.
```

In our analyses we denote by mut(x) the bit string that results from a mutation of x. We denote as x^+ the search point that results from a mutation of x and a subsequent selection. Formally, $y = \text{mut}(x)$ and $x^+ = y$ if $f(y) \geq f(x)$ and $x^+ = x$ otherwise.

For $x = x_1 \dots x_n$ let $Z(x)$ describe the positions of all 0-bits in x, $Z(x) := \{1 \leq i \leq n \mid x_i = 0\}$. By $|x|_0 := |Z(x)|$ we denote the number of 0-bits in x and by $|x|_1 := n - |x|_0$ we denote the number of 1-bits. For $k \in \mathbb{N}$ let $[k] := \{1, 2, \dots, k\}$. For a set $I = \{i_1, i_2, \dots, i_\ell\} \subseteq [n]$ with $i_1 < i_2 < \dots < i_\ell$ we write $x_{|I} := x_{i_1} x_{i_2} \dots x_{i_\ell}$ for the substring of x with the bits selected by I. To simplify notation, we assume that any time we consider some $r \in \mathbb{R}_0^+$ but in fact need some $r' \in \mathbb{N}_0$, then r is silently replaced by $\lfloor r \rfloor$ or $\lceil r \rceil$ as appropriate.

We are interested in the optimization time, defined as the number of mutations until a global optimum is found. For the (1+1) EA, this is an accurate measure of the actual runtime. For bounds on the optimization time, we use common asymptotic notation. Such a bound on the optimization time is called exponential if it is $2^{\Omega(n)}$. We also say that an event A occurs with overwhelming probability (w.o.p.) if $1 - \Pr(A) = 2^{-\Omega(n)}$.

A function f is called linear if it can be written as $f(x) := \sum_{i=1}^{n} a_i x_i$ for weights $a_1, \ldots, a_n \in \mathbb{R}$. The most simple linear function is the function $\text{OneMax}(x) := \sum_{i=1}^{n} x_i = |x|_1$. Another intensively studied linear function is $\text{BinVal}(x) := \sum_{i=1}^{n} 2^{n-i} x_i$. As $2^{n-i} > \sum_{j=i+1}^{n} 2^{n-j}$, the bit value of some bit i dominates the effect of all bits $i+1, \ldots, n$ on the function value. Both functions will later be needed in our construction.

For two search points $x, y \in \{0,1\}^n$, we write $x \leq y$ if $x_i \leq y_i$ holds for all $i \in [n]$. We write $x < y$ if $x \leq y$ and $x \neq y$ hold. We call f a strictly monotonic function (usually called simply monotonic in the following) if for all $x, y \in \{0,1\}^n$ with $x < y$ it holds that $f(x) < f(y)$. Observe that the above condition is equivalent to $f(x) < f(y)$ for all x and y such that x and y only differ in exactly one bit and this bit has value 1 in y. In other words, every mutation that only flips bits with value 0 strictly increases the function value. Clearly, the all 1s bit string 1^n is the unique global optimum for a monotonic function.

Note that every linear function with strictly positive weights is a strictly monotonic function as flipping only 0-bits to 1 strictly increases the fitness. Also recall that every monotonic function is unimodal since for each nonoptimal search point, that is, for each $x \neq 1^n$, we can flip exactly one 0-bit and get a Hamming neighbor y with $f(y) > f(x)$.

If for two arbitrary search points x, y, if neither $x < y$ nor $y < x$ holds, we say that x and y are incomparable. This happens if and only if there are two different bit positions i and j such that $x_i = 0$ but $y_i = 1$ and $x_j = 1$ but $y_j = 0$. Note that monotonicity does not impose any restrictions on the fitness values of x and y. In other words, if f is monotonic, then any of the following cases can occur: $f(x) > f(y)$, $f(x) = f(y)$, or $f(x) < f(y)$. When constructing a monotonic function, we can choose any of the above cases for f, as long as no monotonicity constraint involving other search points is violated. This in particular indicates that the class of monotonic functions contains much more complex functions than linear functions.

3 Runtime Results for Monotonic Functions

For the (1+1) EA, the difficulty of monotonic functions strongly depends on the mutation probability $p(n)$. We are interested in mutation probabilities $p(n) = c/n$ for some constant $c \in \mathbb{R}^+$. If c is a constant with $c < 1$, on average, less than one bit flips in a single mutation. If this is a 1-bit we have $f(x) > f(\text{mut}(x))$ and $x = x^+$ holds. Otherwise, $f(x^+) > f(x)$ holds and we accept this move. This way the number of 0-bits is quickly reduced to 0 and the unique global optimum is found. Using drift analysis, this reasoning can easily be made precise.

THEOREM 1: *Let $c \in \,]0,1[$ be a constant. For every monotonic function, the expected optimization time of the (1+1) EA with mutation probability $p(n) = c/n$ is $\Theta(n \log n)$.*

PROOF: Initially, there are at least $\lfloor n/2 \rfloor$ 0-bits in x with probability at least 1/2. Considering the situation after at most $c^{-1}(n-c)\ln n$ mutations (Droste et al., 2002) the probability that at least one of these bits is never flipped is bounded below by

$$1-\left(1-\left(1-\frac{c}{n}\right)^{c^{-1}(n-c)\ln n}\right)^{\lfloor n/2\rfloor} \geq 1-\left(1-e^{-\ln n}\right)^{\lfloor n/2\rfloor}$$
$$\geq 1-\left(1-\frac{1}{n}\right)^{n/3} \geq 1-e^{-1/3}.$$

This establishes $\Omega(n\log n)$ as lower bound.

For the upper bound we employ multiplicative drift analysis (Doerr, Johannsen, and Winzen, 2010b). We consider the distance measure $d\colon \{0,1\}^n \to \{0,1,\ldots,n\}$ with $d(x) := |x|_0$. Let x denote the current bit string of the (1+1) EA. In order to derive an upper bound on $\mathrm{E}(d(x^+) \mid x)$ we distinguish the two cases $d(x)=d(x^+)$ and $d(x)\neq d(x^+)$. By the law of total probability

$$\mathrm{E}(d(x^+)\mid x)=\Pr(d(x)=d(x^+))\cdot d(x)+\Pr(d(x)\neq d(x^+))\cdot \mathrm{E}(d(x^+)\mid x, d(x)\neq d(x^+))$$

holds. In the case $d(x)\neq d(x^+)$ the bit string x^+ replaces x. This can only be the case if at least one bit flipped from 0 to 1. Each of the remaining $n-1$ bits flips with probability c/n and may increase the distance by 1. This yields

$$\mathrm{E}(d(x^+)\mid x, d(x)\neq d(x^+)) \leq d(x)-1+(n-1)c/n$$
$$\leq d(x)-(1-c)$$

as the upper bound and we obtain

$$\mathrm{E}(d(x^+)\mid x)\leq d(x)-\Pr(d(x)\neq d(x^+))(1-c).$$

A sufficient condition for $d(x)\neq d(x^+)$ is that exactly one of the $d(x)$ bits with value 0 in x flips and all other bits remain unchanged. This event has a probability of

$$\binom{d(x)}{1}\frac{c}{n}\left(1-\frac{c}{n}\right)^{n-1}\leq d(x)\frac{ce^{-c}}{n}$$

and leads to

$$\mathrm{E}(d(x^+)\mid x)\leq d(x)\left(1-\frac{ce^{-c}(1-c)}{n}\right)$$

as the upper bound. Applying the drift theorem (Doerr et al., 2010b) we obtain $(n/(ce^{-c}(1-c)))(1+\ln(n))=O(n\log n)$ as the upper bound on the expected optimization time. □

The proof of the lower bound is not restricted to $c\in\,]0,1[$. For any constant $c>0$, the number of steps considered, $c^{-1}(n-c)\ln n$, is $\Omega(n\log n)$. This implies the following corollary.

COROLLARY 1: *Let $c>0$ be a constant. For every monotonic function, the expected optimization time of the (1+1) EA with mutation probability $p(n)=c/n$ is $\Omega(n\log n)$.*

The proof of the upper bound in Theorem 1 breaks down for $c=1$. In this case, the drift in the number of 1-bits can be bounded pessimistically by a model due to Jansen (2007) where we consider a random process that mutates x to y with mutation probability $p(n)=1/n$ and replaces x by y if either $x\leq y$ holds or we have neither $x\leq y$ nor $y\leq x$ but $|y|_1<|x|_1$ holds.

The model is pessimistic in the following sense. Every mutation that flips only 0-bits to 1-bits is guaranteed to lead to an improvement in the function value for every monotonic function and is accepted in this model, too. For the analysis of the model as well as the (1+1) EA on a monotonic function, drift analysis could be employed using the number of 0-bits as the drift function. With respect to this drift function, the model is more pessimistic than any monotonic function since each mutation that potentially decreases the number of 1-bits in a monotonic function is accepted. This cannot happen for a monotonic function. To see this, consider, for example, $n = 4$ and the following sequence of bit strings: $s_0 = 0111$, $s_1 = 1100$, $s_2 = 0001$, $s_3 = 0011$. In the pessimistic model, we could have s_0, s_1, s_2, s_3, and s_0 as a sequence of current bit strings. This cannot be the case for the (1+1) EA with any monotonic function, since $f(s_2) < f(s_3) < f(s_0)$ holds by definition of monotonicity. Having s_0, s_1 as a sequence of current bit strings implies $f(s_1) \geq f(s_0)$, and since $f(s_0) > f(s_2)$ we cannot have s_2 as the next current bit string. Thus, the pessimistic model allows for cycles that are not possible for the (1+1) EA with any monotonic function.

Using the number of 0-bits as drift function, the worst case model can yield an upper bound for the expected optimization time of the (1+1) EA with mutation probability $p(n) = 1/n$ on monotonic functions. This way, we obtain the upper bound of $O(n^{3/2})$ for $p(n) = 1/n$.

THEOREM 2: *For every monotonic function, the expected optimization time of the (1+1) EA with mutation probability $p(n) = 1/n$ is $O(n^{3/2})$.*

Our main result is that using mutation probability $p(n) = c/n$, where c is a sufficiently large constant, the optimization of monotonic functions can become very difficult for the (1+1) EA. This is the first result where increasing the mutation probability by a constant factor increases the optimization time from polynomial to exponential with overwhelming probability.

THEOREM 3: *For every constant $c \geq 16$ the following holds. For all $n \in \mathbb{N}$, there exists a monotonic function $f\colon \{0,1\}^n \to \mathbb{N}$ and a constant $\kappa > 0$ such that, with probability $1 - 2^{-\Omega(n)}$, the (1+1) EA with mutation probability $p(n) = c/n$ does not optimize f within $2^{\kappa n}$ generations.*

The remainder of this work is devoted to the formal proof of Theorem 3. We first present the construction of such a monotonic function f in the following section, and then prove that it has the desired properties in Section 5.

4 A Difficult to Optimize Monotonic Function

In this section, we describe a monotonic function that is difficult to optimize via a (1+1) EA with mutation probability $p(n) = c/n$, if $c \geq 16$ is constant.

The main idea is the construction of a kind of long path function, similar to the work by Horn et al. (1994). They defined a path of Hamming neighbors (i.e., bit strings differing in exactly one bit) of exponential length. The probability of taking a shortcut by mutation, that is, jumping forward a long distance on the path, is very small, as many bits have to flip simultaneously. All points that are not on the path have an unfavorable fitness, so an evolutionary algorithm is forced to follow the path to the end.

Here, we also have an exponentially long path such that shortcuts can only be taken if a large number of bits flip simultaneously, a very unlikely event. The construction is complicated by the fact that the function needs to be monotonic. Hence, we cannot forbid leaving the path by giving the boundary of the path an unfavorable fitness. We solve this problem, roughly speaking, by implementing the path on a level of bit strings having similar numbers of 1-bits. Monotonicity simply forbids leaving the level to strings having fewer 1-bits. The path is broad in a sense that the algorithm can gather some additional 1-bits without leaving the path. The crucial part of our construction is setting up the function in such a way that, in spite of monotonicity, not too many 1-bits are collected.

Our path will be located at a region where the number of 1-bits is already fairly large. If the mutation probability c/n is large, it is likely that more 1-bits are flipped to 0 than 0-bits are flipped to 1. So, when mutating a point on the path, it is likely that we have a net loss in terms of the number of 1-bits. This effect becomes more pronounced the more 1-bits the mutated search point has. The behavior of the (1+1) EA of course depends on whether such a net loss will be accepted. Monotonicity requires that whenever only 1-bits are flipped to 0, then the fitness must decrease. However, if, say, one 0-bit is flipped to 1 and three 1-bits are flipped to 0, the two search points are incomparable. Hence, even for a monotonic fitness function, such a transition might be accepted. Our long path function is constructed in such a way that operations leading to a net loss of 1-bits when moving to an incomparable offspring are often accepted, while the current search point is on the path. This prevents the algorithm from gathering too many 1-bits and hence leaving the path.

These considerations particularly apply to a subset of bits that we call the window. The precise subset determines the position on the long path; the set of bits in the window changes as the algorithm moves along on the path. More formally, for a subset of indices $B \subseteq [n]$ and for $x \in \{0,1\}^n$ the bits x_i with $i \in B$ are referred to as the window. These indices need not form a block, that is, B can be any subset of $[n]$ and need not necessarily be of the form $B = [\ell_1, \ell_2]$ with $\ell_1, \ell_2 \in [n]$. The bits x_i with $i \notin B$ are outside the window. Inside the window, the function value is given by BinVal. The weights for BinVal are ordered differently for each window in order to avoid correlation between windows. The window is placed such that there is only a small number of 0-bits outside the window. Reducing the number of 0-bits outside causes the window to be moved. This is a likely event that happens frequently. However, we manage to construct an exponentially long sequence of windows with the additional property that in order to come from one window to another one at a large distance (in the sense of this sequence), a large number of bits needs to be flipped simultaneously. Since this is highly unlikely, it is very likely that the sequence of windows is followed, that is, we do not jump from one window to another one at a large distance. Thus, following the path takes, with overwhelming probability, an exponential number of steps. Droste, Jansen, and Wegener (1998) embed the long path into a unimodal function in a way that the (1+1) EA reaches the beginning of the path with probability close to 1. We adopt this technique and extend it to our monotonic function.

The following Lemma 1 defines the sequence of windows of our function by defining the index sets B_i. Concrete values for the upcoming constants β and γ will be given later on in Theorem 4. The property that windows with large distance have large Hamming distance is formally stated as $|i-j| \geq \ell \Rightarrow |B_i \cap B_j| \leq \gamma\ell$ for $\ell = \Theta(n)$ and some constant $\gamma > 0$.

Lemma 1: *Let $\beta, \gamma \in \mathbb{R}$ be constants with $0 < \beta, \gamma < 1$ and $\rho := \beta/(1-\beta) < \gamma < 2\rho$. Let $n \in \mathbb{N}$, $\ell := \beta n$ and $L := \lfloor \exp((\gamma-\rho)^2(1-\beta)n/6) \rfloor$. Finally, let $L' := L - \ell + 1$. Then there exist $b_1, b_2, \ldots, b_L \in [n]$ such that the following holds. Let $B_i := \{b_i, b_{i+1}, \ldots, b_{i+\ell-1}\}$ for all $i \in [L']$. Then*

1. *$|B_i| = \ell$ for all $i \in [L']$,*
2. *$|B_i \cap B_j| \leq \gamma\ell$ for all $i, j \in [L']$ such that $|i-j| \geq \ell$.*

For the proof, we shall use the following lemma from Doerr, Happ, and Klein (in press), which can also be found in Doerr (2011, p. 13).

Lemma 2 (Chernoff Bound for Moderately Independent Random Variables): *Let $X_1, \dots, X_n$ be arbitrary binary random variables. Let $X_1^*, \dots, X_n^*$ be binary random variables that are mutually independent and such that for all i, X_i^* is independent of $X_1, \dots, X_{i-1}$. Assume that for all i and all $x_1, \dots, x_{i-1} \in \{0, 1\}$,*

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

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

$$\Pr\left(\sum_{i=1}^{n} X_i > k\right) \le \Pr\left(\sum_{i=1}^{n} X_i^* > k\right).$$

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

Proof of Lemma 1: The proof invokes the probabilistic method (Alon and Spencer, 2008), that is, we describe a way to randomly choose the b_i that ensures that properties (1) and (2) hold with positive probability. This necessarily implies the existence of such a sequence $b_1, \dots, b_L$.

Let the $b_1, b_2, \dots, b_L$ be chosen uniformly at random subject to condition (1). More precisely, let $b_1 \in [n]$ be chosen uniformly at random. If $b_1, \dots, b_{i-1}$ are already chosen, then choose b_i from $[n] \setminus \{b_{\max\{1,i-\ell\}}, \dots, b_{i-1}\}$ uniformly at random.

Let $i, j \in [L']$ with $i < j$ and $|i - j| \ge \ell$. By definition, the sets B_i and B_j do not share an index in $[L]$. Fix any outcome of B_i. For all $k \in \{0, \dots, \ell - 1\}$ let X_k be the indicator random variable for the event $b_{j+k} \in B_i$. Then $|B_i \cap B_j| = \sum_{k=0}^{\ell-1} X_k$. We have that, conditional on any outcomes of all other b_{j+k-t}, $t \in [j + k - 1]$, the probability $\Pr(X_k = 1)$ that $b_{j+k} \in B_i$ is at most $|B_i|/(n - \ell) = \beta/(1 - \beta) = \rho$.

From this we first conclude that $E(|B_i \cap B_j|) \le \rho\ell$. In addition, we may apply Lemma 2 with the X_i^* being the independent indicator variables taking the value 1 with probability $\beta/(1 - \beta)$, and conclude via a simple Chernoff bound (cf. e.g., Mitzenmacher and Upfal, 2005) that

$$\begin{aligned}\Pr(|B_i \cap B_j| > \gamma\ell) &\le \Pr\left(\sum_{k=0}^{\ell-1} X_k^* > (1 + \tfrac{\gamma-\rho}{\rho}) \cdot \rho\ell\right)\\ &\le \exp(-(\tfrac{\gamma-\rho}{\rho})^2 \rho\ell/3).\end{aligned}$$

Since there are less than $(L')^2$ choices of (i, j), a simple union bound yields

$$\Pr(\exists i, j \in [L'] \colon (|i - j| \ge \ell) \wedge (|B_i \cap B_j| > \gamma\ell)) < (L')^2 \exp(-(\tfrac{\gamma-\rho}{\rho})^2 \rho\ell/3) < 1.$$

□

One technical tool in the definition of the set of difficult monotonic functions is the (random) permutation of vectors that allow us to effectively reduce dependencies between bits. We define the notation for this tool in the following definition.

Definition 1: *Let $\beta, \gamma, \ell, L, L'$, the b_i and B_i be as in Lemma 1. Let $\alpha \in \mathbb{R}$ with $0 < \alpha < \beta$. For $x \in \{0, 1\}^n$ let $\mathcal{B}_x := \{i \in [L'] \mid |Z(x) \setminus B_i| \le \alpha n\}$. Let $i_x^* := \max \mathcal{B}_x$, if $\mathcal{B}_x$ is nonempty. For $i \in [L']$ let $\pi^{(i)}$ be a permutation of B_i. We use the shorthand $\pi^{(i)}(x)$ to denote the vector obtained from permuting the components of $(x_{b_i}, \dots, x_{b_{i+\ell-1}})$ according to $\pi^{(i)}$. Consequently, $\pi^{(i)}(x) = (x_{\pi^{(i)}(b_i)}, \dots, x_{\pi^{(i)}(b_{i+\ell-1})})$.*

The following definition introduces a set of monotonic functions most of which will turn out to be difficult to optimize. The definition assumes the sequence of windows B_i to be given. For $x \in \{0,1\}^n$ we say that some $i \in [L']$ is a potential position in the sequence of windows if the number of 0-bits outside the window B_i is limited by αn, $\alpha > 0$ some constant. We select the largest potential position i as actual position and have the function value for x depend mostly on this position. If no potential position i exists, we have not yet found the path of windows and lead the (1+1) EA toward it. If $i = L'$, that is, the end of the path is reached, the (1+1) EA is led toward the unique global optimum via OneMax.

In addition to the permutations $\pi^{(i)}$ from Definition 1, we define further permutations $\pi'^{(j)}$, $0 \le j \le n$, for the window B_1. These permutations are used to lead the (1+1) EA toward the start of the path and the first window B_1.

DEFINITION 2: *Let $\beta, \gamma, \ell, L, L'$, the b_i and B_i be as in Lemma 1. Let α, $\mathcal{B}_x$, i^*_x be defined as in Definition 1. Let $\pi^{(i)}$ be any permutation of B_i, using the shorthand introduced in Definition 1, and let $\pi'^{(j)}$, $0 \le j \le n$, be any permutation of B_1. Let $\Pi = (\pi'^{(0)}, \dots, \pi'^{(n)}, \pi^{(1)}, \dots, \pi^{(L')})$ denote the sequence of all permutations.*

We define $f_\Pi : \{0,1\}^n \to \mathbb{N}_0$ via

$$f_\Pi(x) := \begin{cases} \left|x_{|[n]\setminus B_1}\right|_1 \cdot 2^n + \mathrm{BinVal}(\pi'^{(|x_{|[n]\setminus B_1}|_1)}(x)), & \text{if } \mathcal{B}_x = \emptyset, \\ i^*_x \cdot 2^{2n} + \mathrm{BinVal}(\pi^{(i^*_x)}(x)), & \text{if } \mathcal{B}_x \neq \emptyset \quad \text{and} \quad i^*_x < L', \\ L \cdot 2^{3n} + |x|_1, & \text{otherwise.} \end{cases}$$

We state one observation concerning the function f_Π that is important in the following. It states that as long as the end of the path of windows is not found, the number of 0-bits outside is not only bounded by αn but equals αn. This property will be used later on to show that the window is moved frequently.

LEMMA 3: *Let $f_\Pi\colon \{0,1\}^n \to \mathbb{N}_0$ be as in Definition 2. Let $x \in \{0,1\}^n$ with $\mathcal{B}_x \neq \emptyset$ and $i^*_x = \max \mathcal{B}_x$. If $i^*_x < L'$, then $\left|Z(x) \setminus B_{i^*_x}\right| = \alpha n$.*

PROOF: By assumption we have $i^*_x < L'$. We consider $B_{i^*_x+1}$ and see that the set coincides with $B_{i^*_x}$ in all but two elements: we have $B_{i^*_x} \setminus B_{i^*_x+1} = \{b_{i^*_x}\}$ and $B_{i^*_x+1} \setminus B_{i^*_x} = \{b_{i^*_x+\ell}\}$. Consequently, $|Z(x) \setminus B_{i^*_x}|$ and $|Z(x) \setminus B_{i^*_x+1}|$ differ by at most one. Thus, $|Z(x) \setminus B_{i^*_x}| < \alpha n$ implies $|Z(x) \setminus B_{i^*_x+1}| \le \alpha n$ and we can replace i^*_x by $i^*_x + 1$. This contradicts $i^*_x = \max \mathcal{B}_x$. We have $|Z(x) \setminus B_{i^*_x}| \le \alpha n$ by definition and thus $|Z(x) \setminus B_{i^*_x}| = \alpha n$ follows. □

Our first main claim is that f_Π is in fact monotonic. This is not difficult, but might not, due to the complicated definition of f_Π, be obvious.

LEMMA 4: *For all Π as above, f_Π is monotonic.*

PROOF: Let $f := f_\Pi$. Let $x \in \{0,1\}^n$ and $j \in [n]$ such that $x_j = 0$. Let $y \in \{0,1\}^n$ be such that $y_k = x_k$ for all $k \in [n] \setminus \{j\}$ and $y_j = 1 - x_j$. That is, y is obtained from x by flipping the jth bit (which is 0 in x) to 1. To prove the lemma, it suffices to show $f(x) < f(y)$.

Let first $\mathcal{B}_x = \emptyset$. If $\mathcal{B}_y \neq \emptyset$ we have $f(x) < n \cdot 2^n + 2^n$ and $f(y) \ge 2^{2n}$ so $f(x) < f(y)$ follows. If $\mathcal{B}_y = \emptyset$ we have either $\left|x_{|[n]\setminus B_1}\right|_1 < \left|y_{|[n]\setminus B_1}\right|_1$ (in case $j \notin B_1$) or $\mathrm{BinVal}(\pi'^{(i)}(x))$ $< \mathrm{BinVal}(\pi'^{(i)}(y))$ (in case $j \in B_1$). In both cases, $f(x) < f(y)$ holds.

Now assume $\mathcal{B}_x \neq \emptyset$ and $i^*_x < L'$. By definition $\mathcal{B}_x \subseteq \mathcal{B}_y$, hence $i^*_y \ge i^*_x$. If $i^*_y = i^*_x$, we conclude with Lemma 3 that $j \in B_{i^*_x}$, and $f(y) > f(x)$ follows from $\mathrm{BinVal}(\pi^{(i^*_y)}(y)) =$

$\text{BinVal}(\pi^{(i_x^*)}(y)) > \text{BinVal}(\pi^{(i_x^*)}(x))$. If $i_y^* > i_x^*$, then $f(y) > f(x)$. In all other cases, $f(x) = L2^{3n} + |x|_1$ and $f(y) = L2^{3n} + |y|_1$, hence $f(y) > f(x)$. □

5 Proof of Theorem 3

By means of the function defined in the previous section (Definition 2), we are now ready to prove Theorem 3. We start with the concrete statement we want to prove.

THEOREM 4: *Consider the (1+1) EA with mutation probability c/n for a constant $c \geq 16$ on the function $f := f_\Pi$ from Definition 2 where Π is chosen uniformly at random and the parameters are chosen according to $\beta := 1/5$, $\gamma := 30/113$, and $\alpha := 3/(400c)$. There is a constant $\kappa > 0$ such that with probability $1 - 2^{-\Omega(n)}$ the (1+1) EA needs at least $2^{\kappa n}$ generations to optimize f.*

This result shows that if f is chosen randomly (according to the construction described), then the (1+1) EA w.o.p. needs an exponential time to find the optimum. Clearly, this implies that there exists a particular function f, that is, a choice of Π, such that the EA faces these difficulties. This is Theorem 3. In fact, there is even an exponential number of functions for which this holds. The parameters α, β, and γ in Theorem 4 were chosen to obtain a small constant in the threshold $16/n$ for the mutation rate.

The proof of Theorem 4 is long and technical. Therefore, we first present an overview over the main proof ideas.

We shall show that both after a typical initialization, when $\mathcal{B}_x = \emptyset$, and afterward, when $\mathcal{B}_x \neq \emptyset$ and $i_x^* < L'$, we have the following situation. There is a window of bits ($B_{i_x^*}$ if i_x^* is defined and B_1 otherwise) such that the fitness of the search points depends mainly on the BinVal function inside the window. Moreover, the fitness is always increased in case the mutation decreases the number of 0-bits outside the window. If $\mathcal{B}_x = \emptyset$ this is due to the term $\left|x_{|[n]\setminus B_1}\right|_1 \cdot 2^n$ in the fitness function and otherwise it is because the current i_x^*-value has increased. The gain in fitness is so large that it dominates any change of the bits inside the window.

We claim that with this construction it is very likely that the current window always contains at least $\eta_{\text{lb}}\beta n$ 0-bits, where η_{lb} is some positive constant. This is proven by showing that in case the number of 0-bits in the window is in the interval $[\eta_{\text{lb}}\beta n, \eta_{\text{ub}}\beta n]$, $0 < \eta_{\text{lb}} < \eta_{\text{ub}} < 1/2$ constant, then there is a tendency (drift) to increase the number of 0-bits again. Applying the drift theorem by Oliveto and Witt (2011) yields that even in an exponential number of generations the probability that the number of 0-bits in the window decreases below $\eta_{\text{lb}}\beta n$ is exponentially small. We first elaborate on why this drift holds and then explain how the lower bound of $\eta_{\text{lb}}\beta n$ 0-bits implies the claim.

If a mutation decreases the number of 0-bits outside the window, the bits inside the window are subject to random, unbiased mutations. Hence, if the number of 0-bits is at most $\eta_{\text{ub}}\beta n$, the expected number of bits flipping from 1 to 0 is larger than the expected number of bits flipping from 0 to 1. Note that a mutation flipping 0-bits to 1 outside the window and flipping 1-bits to 0 inside the window creates an incomparable offspring. If the mutation probability is large enough, the net gain of 0-bits inside the window makes up for the 0-bits lost outside the window. So we have a net gain in 0-bits in expectation, with regard to the whole bit string. Note that the window is moved during such a mutation. As by Lemma 3 the number of 0-bits outside the window is fixed to αn, we have a net gain in 0-bits for the window, regardless of its new position.

In case the number of 0-bits outside the window remains put, acceptance depends on a BinVal instance on the bits inside the window. For BinVal accepting the result of a mutation is completely determined by the flipping bit with the largest weight. In an accepted step, this bit must have flipped from 0 to 1. All bits with smaller weights have no impact on acceptance and therefore are subject to random, unbiased mutations. If, among all bits with smaller weights, there is a sufficiently small rate of 0-bits, more bits will flip from 1 to 0 than from 0 to 1. In this case, we again obtain a net increase in the number of 0-bits in the window, in expectation. Here we again require a large mutation probability since every increase of BinVal implies that one 0-bit has been lost and a surplus of flipping 1-bits has to make up for this loss. This surplus must be generated by flipping 1-bits inside the window that have a small weight. Recall that the window only represents a β-fraction of all bits in the bit string. So, the mutation probability has to be large enough such that the expected number of flipping bits among the mentioned bits is still large enough to make up for the lost bit.

For a fixed BinVal instance the bits tend to develop correlations between bit values and weights over time; bits with large weights are more likely to become 1 than bits with small weights. This development is disadvantageous since the above argument relies on many 1-bits with small weights. In order to break up these correlations we use random instances of BinVal wherever possible. Whenever a new random instance of BinVal is assigned, the bit weights for all bits in the window are reassigned, such that all correlations are lost.

New random instances are applied quickly. If $\mathcal{B}_x = \emptyset$ and, by Lemma 3, also if $i^*_x < L'$ we have exactly αn 0-bits outside the current window and every mutation that flips exactly one of these bits leads to a new BinVal instance. Since this happens with probability $\Omega(1)$, this frequently breaks up correlations and prevents the algorithm from gathering 1-bits at bits inside the window with large BinVal-weights. Pessimistically dealing with bits that have been touched by mutation while optimizing the same BinVal instance, a positive expected increase in the number of 0-bits can be shown.

How does the lower bound of $\eta_{\text{lb}}\beta n$ 0-bits inside the window imply Theorem 4? With overwhelming probability we start with $\mathcal{B}_x = \emptyset$ and at least $\eta_{\text{ub}}\beta n$ 0-bits in the window B_1. We maintain at least $\eta_{\text{lb}}\beta n$ 0-bits in B_1, while the algorithm is encouraged to turn the 0-bits outside of B_1 to 1 quickly. Once the number of 0-bits outside of B_1 has decreased to or below αn, the path has been reached.

The 0-bits in B_1 thereby ensure that the initial i^*_x-value, that is, the initial position on the path, is at most βn. This is because B_1 only has a small overlap to sets B_j that are far further on the path, that is, $j > \beta n$. In general, every two sets B_i, B_j with $|i - j| \geq \ell$ only intersect in at most $\gamma\beta n$ bits. So $\eta_{\text{lb}}\beta n$ 0-bits in B_i imply at least $\eta_{\text{lb}}\beta n - \gamma\beta n$ 0-bits outside of B_j. For j to become the new window, however, at most αn 0-bits outside of B_j are allowed. By choice of α, β, and γ, moving from B_1 to B_j requires a linear number of 0-bits in B_1 to flip to 1 if $j > \beta n$. The described mutation has probability $n^{-\Omega(n)}$. Hence the (1+1) EA finds the start of the long path with overwhelming probability.

The argument on small overlaps also implies that the probability of increasing i^*_x by more than βn in one generation is $n^{-\Omega(n)}$. Hence, even when considering an exponential period of time, with overwhelming probability the (1+1) EA in each generation only makes progress at most βn on the path. As the path has exponential length, the claimed lower bound follows.

In the following, we prove our claim in three steps. We first show that it is very unlikely to take large shortcuts once the path is reached, implying that the algorithm is forced to follow the path (see Section 5.1). Afterward, we make use of drift arguments in order to show that there is always a linear fraction of 0-bits within the current window until the end of the path is reached or an exponential number of iterations have passed (see Section 5.2). The proof of this part is further separated into a part dealing with the case of a moving window and one part where the window stays put. Finally, we show that we hit the beginning of the path starting from a random initialization with overwhelming probability (see Section 5.3). Putting these results together in Section 5.4 then proves Theorem 4.

We remark that the proof of Theorem 4 and the statements above will all be carried out in a parameterized fashion as above. Thus, we actually prove that Theorem 4 holds whenever the following conditions are met.

$$
\begin{gathered}
0 < \eta_{\mathrm{lb}} < \eta_{\mathrm{ub}} < 1/2 \\
0 < \alpha < \beta < \gamma < 1 \\
\eta_{\mathrm{lb}} - \frac{\alpha}{\beta} > \gamma \\
\frac{\beta}{1-\beta} < \gamma < \frac{2\beta}{1-\beta} \\
c\beta > \frac{2 - 2\eta_{\mathrm{ub}}}{1 - 2\eta_{\mathrm{ub}}} \\
\alpha < \frac{1 - 2\eta_{\mathrm{ub}}}{3c}
\end{gathered}
$$

It is easy to check that all conditions are fulfilled by the settings from Theorem 4, that is, whenever $c \geq 16$, $\beta := 1/5$, $\gamma := 30/113$, $\alpha := 3/(400c)$, $\eta_{\mathrm{lb}} := 30/112$, and $\eta_{\mathrm{ub}} := 30/111$.

5.1 Unlikeliness of Shortcuts

We consider the (1+1) EA with mutation probability c/n and say that the (1+1) EA is on level i_x^* if x is the current search point. We also speak of *phase* i_x^* as the random time until the (1+1) EA increases its current level. Note that many phases can be empty. $B_{i_x^*}$ is called the current window of bits in situations where we are looking at a trajectory of these sets and want to emphasize that the bits we are considering might change over time.

The main observation for our analysis is that the current window typically contains at least $\eta_{\mathrm{lb}}\beta n$ 0-bits for some positive constant η_{lb}. This property is maintained even during an exponential number of generations, with overwhelming probability. Under this condition, the probability of increasing the current level i_x^* by a large value is very small. Intuitively speaking, the reason for this is that the sets B_i only have a small intersection and many bits have to change in order to move from $B_{i_x^*}$ to some set B_j with $j \gg i_x^*$. This is made precise in the following lemma.

Lemma 5: *Let $0 < \alpha < \beta < \gamma$ and $0 < \eta_{\mathrm{lb}}$ be constants such that $\eta_{\mathrm{lb}} - \alpha/\beta > \gamma$ and $\beta/(1-\beta) < \gamma < 2\beta/(1-\beta)$. Let f_Π, with respect to α, β, and γ, be constructed as in Definition 2, for arbitrary Π. Let $c > 0$ be a constant and let x be the current search point of the (1+1) EA with mutation probability $p(n) = c/n$ optimizing f_Π. Assume that $\mathcal{B}_x \neq \emptyset$ and $B_{i_x^*}$ contains at least $\eta_{\mathrm{lb}}\beta n$ 0-bits. Then the probability that the (1+1) EA increases the level i_x^* by more than βn in one generation is at most $n^{-\Omega(n)}$.*

Proof: Since $\eta_{\mathrm{lb}}\beta n > \alpha n + \gamma\beta n$, it holds that $B_{i_x^*}$ contains more than $\alpha n + \gamma\beta n$ 0-bits. Recall that $|B_{i_x^*} \cap B_j| \leq \gamma\beta n$ for all $j \geq i_x^* + \beta n$. Thus, there are more than αn 0-bits outside of B_j. This implies, by the definition of $\mathcal{B}_x$, that a necessary condition for increasing i_x^* to any value $j \geq i_x^* + \beta n$ is thus that one mutation decreases the number of 0-bits in $B_{i_x^*}$ to a value below or equal to $\alpha n + \gamma\beta n$. This is a decrease of at least $\eta_{\mathrm{lb}}\beta n - \alpha n - \gamma\beta n =: \kappa n$ bits for some constant $0 < \kappa < 1$. The probability of flipping at least κn bits simultaneously is at most $\binom{n}{\kappa n} \cdot (c/n)^{\kappa n} \leq c^{\kappa n} \cdot 1/(\kappa n)! = n^{-\Omega(n)}$. □

One conclusion from this lemma is that, with overwhelming probability, the (1+1) EA follows the path given by the sets B_i without jumping from one window to another one at a large distance. More precisely, each phase increases the current level by at most βn with overwhelming probability. This will establish the claimed time bound.

5.2 Proving an Invariance Property on the Number of 0-bits in the Current Window

This section deals with the proof of the invariance property on the number of 0-bits in the current window. For this proof we make use of the following drift theorem by Oliveto and Witt (2011).

THEOREM 5: SIMPLIFIED DRIFT THEOREM (OLIVETO AND WITT, 2011): *Let X_t, $t \geq 0$, be the random variables describing a Markov process over a finite state space $S \subseteq \mathbb{R}_0^+$ and denote $\Delta_t(i) := (X_{t+1} - X_t \mid X_t = i)$ for $i \in S$ and $t \geq 0$. Suppose there exists an interval $[a, b]$ in the state space, two constants $\delta, \varepsilon > 0$ and, possibly depending on $I := b - a$, a function $r(I)$ satisfying $1 \leq r(I) = o(I/\log(I))$ such that for all $t \geq 0$ the following two conditions hold:*

1 $\mathrm{E}(\Delta_t(i)) \geq \varepsilon$ *for* $a < i < b$,

2 $\Pr(\Delta_t(i) \leq -j) \leq \frac{r(I)}{(1+\delta)^j}$ *for* $i > a$ *and* $j \in \mathbb{N}_0$.

Then there is a constant $\kappa > 0$ such that for the time $T^* := \min\{t \geq 0 : X_t \leq a \mid X_0 \geq b\}$ it holds that $\Pr(T^* \leq 2^{\kappa I/r(I)}) = 2^{-\Omega(I/r(I))}$.

A prerequisite for this theorem is that the number of 0-bits in the current window increases in expectation when the number of 0-bits is in a certain interval. We choose the interval $[\eta_{\text{lb}}\beta n, (\eta_{\text{lb}} + \eta_{\text{ub}})/2 \cdot \beta n]$, where $0 < \eta_{\text{lb}} < \eta_{\text{ub}} < 1/2$, but establish lower bounds for the drift with respect to a larger interval $[\eta_{\text{lb}}\beta n, \eta_{\text{ub}}\beta n]$. The larger interval will be used later on when proving that after initialization the (1+1) EA finds the start of the path (cf. Lemma 10).

The drift on the number of 0-bits will be bounded from below by positive constants in two cases: either the current level remains fixed in one generation or the current level is increased. We start with the latter case and give a lower bound for the number of 0-bits in the current window. At the end of this section, we apply the above stated drift theorem.

Before we formulate the main statements of this section, we need to introduce some notation. For any x, let $x_B := x_{|B_{i_x^*}}$ denote the substring of x induced by $B_{i_x^*}$, that is, the substring in the current window. Recall that $|x_B|_0$ denotes the number of 0-bits in the current window. That is, $|x_B|_0 = |\{j \in \{b_{i_x^*}, \ldots, b_{i_x^*+\ell-1}\} \mid x_j = 0\}|$. For readability purposes, we write $\left|x_B^+\right|_0$ instead of $\left|(x^+)_B\right|_0$ for the number of 0-bits of x^+ in its window.

5.2.1 Invariance for Sliding Windows

We first consider the case where the current level is increased, that is, a transition from i_x^* to $i_{x^+}^*$ with $i_x^* < i_{x^+}^* < L'$ happens. Note that here we deal with the case $i_x^* \neq i_{x^+}^*$ and thus, $B_{i_x^*} \neq B_{i_{x^+}^*}$ and $x_B^+ \neq x_B$ holds. We show that in this situation we have a drift in the number of 0-bits within the current window that is bounded below by a positive constant. Due to the transition, it is not sufficient to only consider changes within the current window. Furthermore, transitions are often triggered by changes outside the current window. Thus, we assume a form of a global view and take into account both the changes within the current window as well as changes outside the current window. We formalize this within the next lemma.

LEMMA 6: *Let $0 < \alpha < \beta < 1$, $0 < \eta_{\mathrm{ub}} < 1/2$, and c be constants such that $\alpha < \frac{1-2\eta_{\mathrm{ub}}}{3c}$ and $c\beta > \frac{2-2\eta_{\mathrm{ub}}}{1-2\eta_{\mathrm{ub}}}$. Let n be sufficiently large and let f, with respect to α and β, be constructed as in Theorem 4. Let x be the current search point of the (1+1) EA with mutation probability $p(n) = c/n$ maximizing f. We denote by $\bar{A}$ the event that a transition from level i_x^* to $i_{x^+}^*$ with $i_x^* < i_{x^+}^* < L'$ occurs in an iteration of the (1+1) EA maximizing f. Assume $|x_B|_0 \le \eta_{\mathrm{ub}}\beta n$.*

Then there is a constant $\delta > 0$ such that the drift in the number of 0-bits is at least δ, that is, $\mathrm{E}(|x_B^+|_0 - |x_B|_0 \mid \bar{A}) \ge \delta$.

PROOF: Let $\overline{B}_{i_x^*} = [n] \backslash B_{i_x^*}$, the indices not contained in the current window, and $x_{\overline{B}} := x_{|\overline{B}_{i_x^*}}$ the corresponding induced substring of x. Analogously, we define $x_{\overline{B}}^+ := x_{|\overline{B}_{i_{x^+}^*}}$. Due to Lemma 3, we have $|x_{\overline{B}}|_0 = |x_{\overline{B}}^+|_0 = \alpha n$.

The main part of the proof is to derive a lower bound on $\mathrm{E}(|x_B^+|_0 \mid \bar{A})$. Afterward we show that this bound together with the given prerequisites on α, β, c, and $|x_B|_0$ yields a positive drift in the number of 0-bits.

It is easy to see that, conditional on $\bar{A}$, the expected number of 0-bits in the new window $B_{i_{x^+}^*}$ after a transition from i_x^* to $i_{x^+}^*$ can be derived as the difference of the expected number of 0-bits in the current window $B_{i_x^*}$ after mutation and the expected amount of 0-bits lost outside the current window due to mutation:

$$\mathrm{E}\left(|x_B^+|_0 \mid \bar{A}\right) = \mathrm{E}\left(|\mathrm{mut}(x_B)|_0 \mid \bar{A}\right) - \mathrm{E}\left(|x_{\overline{B}}|_0 - |\mathrm{mut}(x_{\overline{B}})|_0 \mid \bar{A}\right). \tag{1}$$

We derive bounds for both parts of Equation (1) separately. We start with an upper bound on the expected number of 0-bits in the current window after mutation, that is $\mathrm{E}\left(|\mathrm{mut}(x_B)|_0 \mid \bar{A}\right)$ by the following case distinction.

In the first case, the transition happens independently of the change in the window. This case happens with probability $\Omega(1)$ as a 1-bit mutation of one of the αn 0-bits outside the current window suffices. In this situation, the expected number of 0-bits in the window is independent of $\bar{A}$ and thus, can be easily calculated as follows.

$$\begin{aligned}
\mathrm{E}\left(|\mathrm{mut}(x_B)|_0\right) &= \left(1 - \frac{c}{n}\right)|x_B|_0 + \frac{c}{n}|x_B|_1 \\
&= |x_B|_0 - \frac{c}{n}|x_B|_0 + \frac{c}{n}|x_B|_1 \\
&= |x_B|_0 + \frac{c}{n}\left(\beta n - 2|x_B|_0\right) \\
&= |x_B|_0 + c\beta - \frac{2c\,|x_B|_0}{n}.
\end{aligned}$$

For the second case, that is, if the mutation within the current window has influence on the transition performed, we have to be more careful, as the expected number of 0-bits within the window is no longer independent of $\bar{A}$. However, before the mutation the leftmost bit in the current window is 0. Otherwise, the next window position would also be a potential, higher window position. This contradicts the definition of i_x^*. If this leftmost 0-bit is flipped, a transition is performed. Furthermore, it is necessary to flip this leftmost 0-bit if the mutation within the window is supposed to have influence on the transition performed. The probability of flipping this bit is c/n and thus the probability for this case is at most c/n.

We bound the contribution of this case in a pessimistic way. Similar to Lemma 5, we see that the number of bits flipping in one single iteration is at most $O(\log n)$ with probability $1 - n^{-\omega(1)}$. Furthermore, the contribution is at most βn otherwise. Altogether, this yields a contribution to the expected value of at most

$$\frac{c}{n} \cdot \left((1 - n^{-\omega(1)}) \cdot \log n + n^{-\omega(1)} \cdot \beta n \right) = O\left(\frac{\log n}{n} \right),$$

leading to the following lower bound on the expected number of 0-bits within the current window after mutation.

$$\mathrm{E}\left(|\mathrm{mut}(x_B)|_0 \mid \bar{A} \right) \geq |x_B|_0 + c\beta - \frac{2c\,|x_B|_0}{n} - O\left(\frac{\log n}{n} \right). \tag{2}$$

The second part of Equation (1), that is, the expected loss of 0-bits outside the current window due to mutation, is more difficult. For the sake of readability, let $k := \left|x_{\overline{B}}\right|_0 - \left|\mathrm{mut}(x_{\overline{B}})\right|_0$ denote the loss of 0-bits outside the current window due to mutation. We are then searching for $\mathrm{E}(k \mid \bar{A})$. We distinguish two cases. If $k > 0$ we definitely observe a transition and accept the new search point. If $k \leq 0$ a transition does not necessarily occur. For $k < 0$ the new search point is only accepted if this is the case. For $k = 0$ the search might also be accepted depending on the changes within the current window. We see that $\mathrm{E}(k) \leq \mathrm{E}(k \mid \bar{A}) \leq \mathrm{E}(k \mid k > 0)$ holds.

Let Z_0 be the number of 0-bits flipping to 1 and Z_1 the number of 1-bits flipping to 0. This yields $\mathrm{E}(k \mid k > 0) = \mathrm{E}(Z_0 - Z_1 \mid Z_0 > Z_1)$. Moreover, we observe that

$$\begin{aligned}
&\sum_{j=i+1}^{\alpha n} j \Pr(Z_0 = j) \\
&\quad = (i+1)\Pr(Z_0 = i+1) + (i+1)\Pr(Z_0 = i+2) + \cdots + (i+1)\Pr(Z_0 = \alpha n) \\
&\qquad\qquad\qquad\qquad + \Pr(Z_0 = i+2) + \cdots + \Pr(Z_0 = \alpha n) \\
&\qquad\qquad\qquad\qquad + \Pr(Z_0 = i+3) + \cdots + \Pr(Z_0 = \alpha n) \\
&\qquad\qquad\qquad\qquad\qquad \ddots \\
&\qquad\qquad\qquad\qquad\qquad\qquad + \Pr(Z_0 = \alpha n) \\
&= (i+1)\Pr(Z_0 > i) + \Pr(Z_0 > i+1) + \Pr(Z_0 > i+2) + \cdots + \Pr(Z_0 > \alpha n - 1) \\
&= (i+1)\Pr(Z_0 > i) + \sum_{j=i+1}^{\alpha n} \Pr(Z_0 > j)
\end{aligned} \tag{3}$$

holds. Then with $\left|x_{\overline{B}}\right|_0 = \alpha n$ and $\left|x_{\overline{B}}\right|_1 = (1 - \alpha - \beta)n$ we can rewrite the expected value sought as follows.

$$\begin{aligned}
\mathrm{E}(Z_0 - Z_1 \mid Z_0 > Z_1) &= \sum_{i=0}^{(1-\alpha-\beta)n} \Pr(Z_1 = i) \cdot \mathrm{E}(Z_0 - Z_1 \mid Z_0 > Z_1 \text{ and } Z_1 = i) \\
&= \sum_{i=0}^{(1-\alpha-\beta)n} \Pr(Z_1 = i) \cdot (\mathrm{E}(Z_0 \mid Z_0 > i) - \mathrm{E}(Z_1 \mid Z_0 > Z_1 \text{ and } Z_1 = i)) \\
&= \sum_{i=0}^{(1-\alpha-\beta)n} \Pr(Z_1 = i) \cdot (\mathrm{E}(Z_0 \mid Z_0 > i) - i)
\end{aligned}$$

$$
\begin{aligned}
&= \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \Pr(Z_1 = i) \cdot \left(\sum_{j=0}^{\alpha n} j \cdot \Pr(Z_0 = j \mid Z_0 > i) - i \right) \\
&= \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \Pr(Z_1 = i) \cdot \left(\sum_{j=0}^{\alpha n} j \cdot \frac{\Pr(Z_0 = j \text{ and } Z_0 > i)}{\Pr(Z_0 > i)} - i \right) \\
&= \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \Pr(Z_1 = i) \cdot \left(\frac{\sum_{j=i+1}^{\alpha n} j \cdot \Pr(Z_0 = j)}{\Pr(Z_0 > i)} - i \right) \\
&\overset{(3)}{=} \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \Pr(Z_1 = i) \cdot \left(\frac{(i+1) \cdot \Pr(Z_0 > i) + \sum_{j=i+1}^{\alpha n} \Pr(Z_0 > j)}{\Pr(Z_0 > i)} - i \right) \\
&= \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \Pr(Z_1 = i) \cdot \left(1 + \frac{\sum_{j=i+1}^{\alpha n} \Pr(Z_0 > j)}{\Pr(Z_0 > i)} \right). \qquad (4)
\end{aligned}
$$

It is easy to see the following estimates for the probabilities used above.

$$
\begin{aligned}
\Pr(Z_0 > j) &\leq \binom{\alpha n}{j+1} \left(\frac{c}{n}\right)^{j+1}, \\
\Pr(Z_0 > i) &\geq \binom{\alpha n}{i+1} \left(\frac{c}{n}\right)^{i+1} \left(1 - \frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}, \\
\Pr(Z_1 = i) &= \binom{(1-\alpha-\beta)n}{i} \left(\frac{c}{n}\right)^{i} \cdot \left(1 - \frac{c}{n}\right)^{(1\text{-}\alpha\text{-}\beta)n\text{-}i}.
\end{aligned}
$$

Plugging these inequalities into Equation (4) yields the following expression for the expected loss of 0-bits outside the current window due to mutation.

$$
\begin{aligned}
\mathrm{E}(k \mid \bar{A}) \leq{} & \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \binom{(1-\alpha-\beta)n}{i} \cdot \left(\frac{c}{n}\right)^{i} \cdot \left(1 - \frac{c}{n}\right)^{(1\text{-}\alpha\text{-}\beta)n\text{-}i} \\
& \cdot \left(1 + \frac{\sum_{j=i+1}^{\alpha n} \binom{\alpha n}{j+1} \left(\frac{c}{n}\right)^{j+1}}{\binom{\alpha n}{i+1} \left(\frac{c}{n}\right)^{i+1} \left(1 - \frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}} \right) \qquad (5) \\
={} & 1 + \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n} \binom{(1-\alpha-\beta)n}{i} \cdot \left(\frac{c}{n}\right)^{i} \cdot \left(1 - \frac{c}{n}\right)^{(1\text{-}\alpha\text{-}\beta)n\text{-}i} \\
& \cdot \frac{\sum_{j=i+1}^{\alpha n} \binom{\alpha n}{j+1} \left(\frac{c}{n}\right)^{j+1}}{\binom{\alpha n}{i+1} \left(\frac{c}{n}\right)^{i+1} \left(1 - \frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}. \qquad (6)
\end{aligned}
$$

We start with the second part of this term and derive the following lower bound.

$$\begin{aligned}
\frac{\sum_{j=i+1}^{\alpha n}\binom{\alpha n}{j+1}\left(\frac{c}{n}\right)^{j+1}}{\binom{\alpha n}{i+1}\left(\frac{c}{n}\right)^{i+1}\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}} &= \frac{\sum_{j=i+1}^{\alpha n}\frac{(\alpha n)!}{(j+1)!(\alpha n-j-1)!}\left(\frac{c}{n}\right)^{j+1}\frac{(i+1)!(\alpha n-i-1)!}{(\alpha n)!}\left(\frac{n}{c}\right)^{i+1}}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}} \\
&= \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\sum_{j=i+1}^{\alpha n}\left(\frac{c}{n}\right)^{j\text{-}i}\frac{(i+1)!}{(j+1)!}\frac{(\alpha n-i-1)!}{(\alpha n-j-1)!} \\
&= \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\sum_{j=i+1}^{\alpha n}\left(\frac{c}{n}\right)^{j\text{-}i}\cdot\frac{\alpha n-i-1}{i+2}\cdot\frac{\alpha n-i-2}{i+3}\cdot\ldots\cdot\frac{\alpha n-j}{j+1} \\
&\le \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\sum_{j=i+1}^{\alpha n}\left(\frac{c}{n}\right)^{j\text{-}i}\left(\frac{\alpha n}{i+1}\right)^{j\text{-}i} = \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\sum_{j=i+1}^{\alpha n}\left(\frac{c\alpha}{i+1}\right)^{j\text{-}i} \\
&= \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\sum_{j=1}^{\alpha n\text{-}i}\left(\frac{c\alpha}{i+1}\right)^{j} \le \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\cdot\frac{\frac{c\alpha}{i+1}}{1-\frac{c\alpha}{i+1}} \\
&\le \frac{1}{\left(1-\frac{c}{n}\right)^{\alpha n\text{-}i\text{-}1}}\cdot\frac{c\alpha}{1-c\alpha}.
\end{aligned}$$

Remember that we assume $\alpha < (1-2\eta_{\text{ub}})/(3c)$ and thus $c\alpha < 1/3$ holds. Therefore, we can further simplify the above inequality by using the simple estimate

$$\begin{aligned}
\left(1-\tfrac{c}{n}\right)^{\alpha n\text{-}i\text{-}1} \ge \left(1-\tfrac{c}{n}\right)^{\alpha n} &= \left(1-\tfrac{c}{n}\right)^{\left(\frac{n}{c}\text{-}1\right)c\alpha}\left(1-\tfrac{c}{n}\right)^{c\alpha} \\
&\ge e^{\text{-}c\alpha}\left(1-\tfrac{c}{n}\right)^{c\alpha} = \left(\frac{1-\frac{c}{n}}{e}\right)^{c\alpha}.
\end{aligned}$$

Plugging all this into Equation (6) yields an upper bound on the expected loss of 0-bits.

$$\begin{aligned}
\mathrm{E}(k \mid \bar{A}) &\le 1 + \sum_{i=0}^{(1\text{-}\alpha\text{-}\beta)n}\binom{(1-\alpha-\beta)n}{i}\left(\frac{c}{n}\right)^{i}\left(1-\frac{c}{n}\right)^{(1\text{-}\alpha\text{-}\beta)n\text{-}i}\left(\frac{e}{1-\frac{c}{n}}\right)^{c\alpha}\frac{c\alpha}{1-c\alpha} \\
&= 1 + \left(\frac{e}{1-\frac{c}{n}}\right)^{c\alpha}\frac{c\alpha}{1-c\alpha}. \qquad (7)
\end{aligned}$$

We are now able to put the results from Equations (2) and (7) together to get a lower bound on Equation (1).

$$\begin{aligned}
\mathrm{E}(\left|x_B^+\right|_0 \mid \bar{A}) &= \mathrm{E}(|\mathrm{mut}(x_B)|_0 \mid \bar{A}) - \mathrm{E}(\left|x_{\overline{B}}\right|_0 - \left|\mathrm{mut}(x_{\overline{B}})\right|_0 \mid \bar{A}) \\
&\ge |x_B|_0 + c\beta - \frac{2c\,|x_B|_0}{n} - O\left(\frac{\log n}{n}\right) - \left(1+\left(\frac{e}{1-\frac{c}{n}}\right)^{c\alpha}\frac{c\alpha}{1-c\alpha}\right).
\end{aligned}$$

With $|x_B|_0 \le \eta_{\text{ub}}\beta n$, this yields the following lower bound on the drift in the number of 0-bits.

$$\begin{aligned}
\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid \bar{A}) &\ge c\beta - \frac{2c\,|x_B|_0}{n} - O\left(\frac{\log n}{n}\right) - \left(1+\left(\frac{e}{1-\frac{c}{n}}\right)^{c\alpha}\frac{c\alpha}{1-c\alpha}\right) \\
&\ge c\beta(1-2\eta_{\text{ub}}) - O\left(\frac{\log n}{n}\right) - 1 - \left(1-\frac{c}{n}\right)^{\text{-}c\alpha}\frac{e^{c\alpha}c\alpha}{1-c\alpha}.
\end{aligned}$$

Clearly,

$$\left(1-\frac{c}{n}\right)^{-c\alpha} = \left(1+\frac{c}{n-c}\right)^{c\alpha} = 1+O(1/n).$$

As the factor $(e^{c\alpha}c\alpha)/(1-c\alpha)$ is constant, we have $O(1/n)\cdot(c\alpha)/(1-c\alpha) = O(1/n)$ and thus, this term can be absorbed by the $O((\log n)/n)$-term. This results in the lower bound

$$\begin{aligned} & c\beta(1-2\eta_{\text{ub}}) - O\left(\frac{\log n}{n}\right) - 1 - \frac{e^{c\alpha}c\alpha}{1-c\alpha} \\ \geq\ & c\beta(1-2\eta_{\text{ub}}) - O\left(\frac{\log n}{n}\right) - 1 - \frac{3e^{1/3}}{2}\cdot c\alpha, \end{aligned} \tag{8}$$

where we have again used $c\alpha < 1/3$. Combining the preconditions $c\beta > (2-2\eta_{\text{ub}})/(1-2\eta_{\text{ub}})$ and $\alpha < (1-2\eta_{\text{ub}})/(3c)$ implies that

$$c\beta > \frac{1+(1-2\eta_{\text{ub}})}{1-2\eta_{\text{ub}}} > \frac{1+3c\alpha}{1-2\eta_{\text{ub}}}\ .$$

Plugging this into Equation (8) yields

$$1+3c\alpha - O\left(\frac{\log n}{n}\right) - 1 - \frac{3e^{1/3}}{2}\cdot c\alpha = c\alpha\left(3-\frac{3e^{1/3}}{2}\right) - O\left(\frac{\log n}{n}\right).$$

This is bounded from below by a positive constant δ for sufficiently large values of n, which concludes the proof. □

5.2.2 Invariance for Nonsliding Windows

In the following, we deal with the case $i^*_x = i^*_{x^+}$. We show that, whenever the number of 0-bits in the current window is in the interval $[\eta_{\text{lb}}\beta n, \eta_{\text{ub}}\beta n]$, we observe a drift toward more 0-bits. This is formalized in the following lemma.

LEMMA 7: *Let $0<\alpha<\beta<1$, $0<\eta_{\text{lb}}<\eta_{\text{ub}}<1/2$, and c be constants such that $c\beta > \frac{2-2\eta_{\text{ub}}}{1-2\eta_{\text{ub}}}$. Let n be sufficiently large and let f, with respect to α and β, be constructed as in Definition 2. Let x be the current search point of the (1+1) EA with mutation probability $p(n)=c/n$ maximizing f. Assume $|x_B|_0 \in [\eta_{\text{lb}}\beta n, \eta_{\text{ub}}\beta n]$. We denote by A the event that the (1+1) EA maximizing f and starting in x does not leave the current level, that is, $i^*_x = i^*_{x^+}$. Then the following two statements hold.*

1. *For every constant $\varepsilon>0$ the number of different bits that are flipped during phase i^*_x is at most $2\varepsilon\beta cn$, with probability $1-\exp(-\Omega(n))$.*
2. *For small enough $\varepsilon>0$, assuming that the event from (1) holds, there exists a constant $\delta>0$ such that the drift in the number of 0-bits $\mathrm{E}(|x^+_B|_0 - |x_B|_0 \mid A)$ is at least δ.*

The proof of this lemma will heavily depend on the drift in the number of 0-bits induced by the random BinVal within the current window. In the proof of Lemma 7, we will have to deal with variable lengths of the considered bit string. Therefore, the following auxiliary lemma is formulated for a bandwidth of possible bit string lengths. One precondition is that bit weights of BinVal are assigned uniformly at random. This is the case right after a new BinVal instance has been set.

LEMMA 8: *Let $0\leq\varepsilon<\beta<1$, $0<\eta_{\text{lb}}<\eta_{\text{ub}}<1/2$, and c be constants such that*

$$c(\beta-\varepsilon) - 2\eta_{\text{ub}}c\beta > 2-2\eta_{\text{ub}}\frac{\beta}{\beta-\varepsilon} \tag{9}$$

and

$$c(\beta - \varepsilon) \geq 1 \; . \tag{10}$$

Consider the (1+1) EA with mutation probability $p(n) = c/n$ maximizing a BinVal *function on $u \geq \beta n - \varepsilon n$ bits where the weight of the bits is chosen uniformly at random, without replacement, from $\{2^0, 2^1, \ldots, 2^{\beta n\text{-}1}\}$. Let $\tilde{x}$ denote the current search point. If $|\tilde{x}|_0 \in [\eta_{\mathrm{lb}} u, \eta_{\mathrm{ub}} \beta n]$ then there exists a constant $\tilde{\delta} > 0$ such that the drift in the number of* 0*-bits is at least $\tilde{\delta}$, that is,* $\mathrm{E}(\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0) \geq \tilde{\delta}$.

In order to prevent confusion, let us remark that the expectation is taken both with respect to the random assignment of the function weights as well as with respect to the position of the 0-bits of $\tilde{x}$.

Proof of Lemma 8: As a first simple observation, let us recall the following. Whenever $\tilde{x}^+ = \tilde{x}$, it holds that $\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0 = 0$. Thus, we are only interested in the case $\tilde{x}^+ \neq \tilde{x}$. Note that in this case, the construction of BinVal implies that the bit with the largest weight is one that flips from 0 to 1, as the (1+1) EA would otherwise not accept $\mathrm{mut}(\tilde{x})$ as a new search point. For all other bits that are being flipped in this iteration, the direction of the flipping bit (i.e., whether the bit itself is a 0-bit flipping to 1 or a 1-bit flipping to 0) is random and does only depend on the shares of 0- and 1-bits. This will be formalized in the following.

For readability purposes, let us introduce the following notations. For every $k \in \{0, \ldots, u\}$ we denote by p_k the probability that the (1+1) EA flips exactly k bits in the mutation step. Clearly, $p_k = \binom{u}{k}(\frac{c}{n})^k(1-\frac{c}{n})^{u\text{-}k}$ for $k \geq 1$ and $p_0 = (1-\frac{c}{n})^u$.

Let us, for the moment, assume that exactly $k \geq 1$ bits are being flipped and let us consider the substring of the flipping bits only. If we remove from the substring the bit with the largest weight (which flips from 0 to 1), we get that the expected number of 0-bits in this reduced substring equals $(k-1)\frac{|\tilde{x}|_0 - 1}{u-1}$. Analogously, the expected number of 1-bits in the bit string equals $(k-1) - (k-1)\frac{|\tilde{x}|_0-1}{u-1}$. Recall that we have chosen the bits to exactly those which are being flipped in the mutation step. We thus obtain for this specific setting that the expected difference of $\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0$ equals

$$\begin{aligned}&\left((k-1) - (k-1)\frac{|\tilde{x}|_0 - 1}{u-1}\right) - (k-1)\frac{|\tilde{x}|_0 - 1}{u-1} - 1\\ &= k\left(1 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right) - \left(2 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right).\end{aligned}$$

Now, for any such k, it holds that $\mathrm{E}(\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0 \mid k \text{ bits flip})$ equals the probability that the flipping bit with the largest weight flips from 0 to 1 (which occurs with probability $\frac{|\tilde{x}|_0}{u}$) times the drift conditional on k bit flips. The latter equals $k\left(1 - 2\frac{|\tilde{x}|_0-1}{u-1}\right) - \left(2 - 2\frac{|\tilde{x}|_0-1}{u-1}\right)$ as outlined above.

Combining these observations, we gain

$$\mathrm{E}(\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0) \;=\; \sum_{k=1}^{u} p_k \frac{|\tilde{x}|_0}{u}\left[k\left(1 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right) - \left(2 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right)\right].$$

Clearly, $\sum_{k=0}^{u} p_k = 1$ as we are dealing with a probability distribution. Thus, $\sum_{k=1}^{u} p_k = 1 - (1-\frac{c}{n})^u$. On the other hand, $\sum_{k=1}^{u} p_k k = \frac{c}{n}u$ as this sum equals the expected number of bit flips. By the choice of the parameters we have $\sum_{k=1}^{u} p_k k \geq c(\beta - \varepsilon)$. This yields

$$\begin{aligned}
\mathrm{E}(\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0) &\geq \frac{|\tilde{x}|_0}{u}\left[\left(1 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right)c(\beta - \varepsilon) - \left(2 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right)\left(1 - \left(1 - \frac{c}{n}\right)^u\right)\right] \\
&\geq \eta_{\mathrm{lb}}\left[\left(1 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right)c(\beta - \varepsilon) - \left(2 - 2\frac{|\tilde{x}|_0 - 1}{u-1}\right)\right].
\end{aligned}$$

As $c(\beta - \varepsilon) \geq 1$ by Equation (10), we can use the estimate $\frac{|\tilde{x}|_0 - 1}{u-1} \leq \frac{|\tilde{x}|_0}{u} \leq \eta_{\mathrm{ub}}\frac{\beta n}{u} \leq \eta_{\mathrm{ub}}\frac{\beta}{\beta - \varepsilon}$, leading to

$$\begin{aligned}
\mathrm{E}(\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0) &\geq \eta_{\mathrm{lb}}\left[\left(1 - 2\eta_{\mathrm{ub}}\frac{\beta}{\beta - \varepsilon}\right)c(\beta - \varepsilon) - \left(2 - 2\eta_{\mathrm{ub}}\frac{\beta}{\beta - \varepsilon}\right)\right] \\
&= \eta_{\mathrm{lb}}\left[c(\beta - \varepsilon) - 2\eta_{\mathrm{ub}}c\beta - 2 + 2\eta_{\mathrm{ub}}\frac{\beta}{\beta - \varepsilon}\right]. \qquad (11)
\end{aligned}$$

By Equation (9), $\mathrm{E}(\left|\tilde{x}^+\right|_0 - |\tilde{x}|_0)$ is bounded from below by some positive constant $\tilde{\delta}$. □

We can now easily deduce Lemma 7.

Proof of Lemma 7: Let us assume that event A (as defined in the statement) holds. That is, the acceptance of the mutated bit string $\mathrm{mut}(x)$ is fully determined by the random BinVal within the current window. Thus, we can restrict our attention to the current window.

Let us begin with proving the first claim. For this purpose, let $\varepsilon > 0$ be a constant. We prove an auxiliary claim stating that with probability 1-exp$(-\Omega(n))$ the time $T_{i_x^*}$ until the (1+1) EA exits level i_x^* is at most εn. That is, we can assume that phase i_{x^*} does not take longer than εn steps. We then show how to derive the original claim.

By construction, the (1+1) EA exits level i_x^* if exactly one of the αn 0-bits outside the current window is being flipped. Thus, the probability $\Pr(i_x^* \neq i_{x^+}^*)$ of exiting the current level in one step is at least $\alpha n \cdot \frac{c}{n} \cdot (1 - \frac{c}{n})^{n\text{-}1} \geq \alpha c e^{\text{-}c}(1 - \frac{c}{n})^{c\text{-}1}$. It follows that the probability of not exiting level i_x^* in εn steps is at most $(1 - \alpha c e^{\text{-}c}(1 - \frac{c}{n})^{c\text{-}1})^{\varepsilon n} \leq \exp(-\alpha c e^{\text{-}c}(1 - \frac{c}{n})^{c\text{-}1}\varepsilon n) = \exp(-\Omega(n))$.

Now, the expected number of bits that have been flipped in εn steps is at most $\varepsilon n \cdot \beta n \cdot \frac{c}{n} = \varepsilon \beta c n$. We apply a standard Chernoff bound, and obtain that the probability that more than $2\varepsilon\beta c n$ bits are being flipped in εn steps is at most $\exp(-\frac{1}{3}\varepsilon\beta c n) = \exp(-\Omega(n))$.

We continue with the second claim. Let us assume that no more than $2\varepsilon\beta c n$ bits are being flipped during phase i_x^*. Note that we can conclude the following. The probability of flipping in the current iteration a bit that has already been flipped in a former iteration of phase i_x^* is at most $2\varepsilon\beta c n \cdot \frac{c}{n} = 2\varepsilon\beta c^2$. That is, $\Pr(G \mid A) \leq 2\varepsilon\beta c^2$, where we denote by G the event that in the current iteration, the (1+1) EA flips a bit that has already been flipped in a former iteration of phase i_x^*. Clearly,

$$\begin{aligned}
\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A) = {} & \Pr(G \mid A)\,\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A \wedge G) \\
& + (1 - \Pr(G \mid A))\,\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A \wedge \bar{G}),
\end{aligned}$$

with $\bar{G}$ denoting the complementary event of G. Now, whenever G occurs, we adopt a worst case view by assuming that all bits flip in the wrong direction, that is, from

0 to 1. For this purpose, let us, for the moment, assume that G holds. In this case, at least one bit flips and we assume very pessimistically that each of the flipping bits reduces the number of 0-bits by 1. Note that, given that one bit flips, the expected number of total bit flips in the current window equals $1 + \frac{c}{n}(\beta n - 1) < 1 + c\beta$. Thus, we can bound $\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A \wedge G)$ from below by $-1 - c\beta$. That is, under our assumption, it holds that

$$\Pr(G \mid A)\,\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A \wedge G) \geq 2\varepsilon\beta c^2(-1 - c\beta)\,.$$

We now need to give bounds for the second summand. For this purpose, we apply Lemma 8. As we are conditioning on $\bar{G}$, we apply the auxiliary lemma with u denoting the number of bits that have not been flipped in any former iteration of phase i_x^*. Furthermore, we are only interested in the substring $\tilde{x}$ of $x_{|B}$ consisting of these u yet unflipped bits. As we have seen in the first part of this proof, with probability $1 - \exp(-\Omega(n))$ it holds that $u \geq \beta n - 2\varepsilon\beta c n$. Also recall that $c\beta > \frac{2-2\eta_{\mathrm{ub}}}{1-2\eta_{\mathrm{ub}}}$ or, equivalently, $c\beta - 2\eta_{\mathrm{ub}} c\beta > 2 - 2\eta_{\mathrm{ub}}$. As β, c, and η_{ub} are constants and the inequality is strict, we can find some small enough $\varepsilon > 0$ such that the stronger statement

$$c(\beta - \varepsilon) - 2\eta_{\mathrm{ub}} c\beta > 2 - 2\eta_{\mathrm{ub}}\frac{\beta}{\beta - \varepsilon}$$

holds, fulfilling the precondition in Equation (9) in Lemma 8. In addition, $c\beta > \frac{2-2\eta_{\mathrm{ub}}}{1-2\eta_{\mathrm{ub}}} \geq 2$, hence $c(\beta - \varepsilon) \geq 1$ for small enough ε, fulfilling the precondition in Equation (10) in Lemma 8. Invoking Lemma 8 yields $\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A \wedge \bar{G}) \geq \tilde{\delta}$ for some positive constant $\tilde{\delta}$.

Altogether we obtain that

$$\mathrm{E}(\left|x_B^+\right|_0 - |x_B|_0 \mid A) \geq 2\varepsilon\beta c^2(-1 - c\beta) + (1 - 2\varepsilon\beta c^2)\tilde{\delta}\,.$$

Last, we observe that we can choose ε small enough such that this term can be bounded from below by some positive constant δ, as claimed. □

5.2.3 Applying the Drift Theorem

Finally, we prove the claimed invariance property.

LEMMA 9: *Let $0 < \alpha < \beta < 1$, $0 < \eta_{\mathrm{lb}} < \eta_{\mathrm{ub}} < 1/2$, γ, and c be constants such that $\beta/(1-\beta) < \gamma < 2\beta/(1-\beta)$, $c\beta > \frac{2-2\eta_{\mathrm{ub}}}{1-2\eta_{\mathrm{ub}}}$, and $\alpha < \frac{1-2\eta_{\mathrm{ub}}}{3c}$. Let f_Π, with respect to α and β, be constructed as in Definition 2, for Π chosen uniformly at random.*

Assume that for the current search point x of the (1+1) EA with mutation probability $p(n) = c/n$ it holds $\mathcal{B}_x \neq \emptyset$ and the current window contains at least $(\eta_{\mathrm{lb}} + \eta_{\mathrm{ub}})/2 \cdot \beta n$ 0-bits. There is a constant $\kappa > 0$ such that with probability $1 - 2^{-\Omega(n)}$ in the following $2^{\kappa n}$ generations the (1+1) EA always has at least $\eta_{\mathrm{lb}}\beta n$ 0-bits in the current window or the end of the path is reached.

PROOF: First, observe that the event described in the first statement of Lemma 7 occurs with probability $1 - 2^{-\Omega(n)}$. By the union bound, the probability that the event occurs within $2^{\kappa n}$ phases is still $1 - 2^{-\Omega(n)}$ if $\kappa > 0$ is a sufficiently small constant.

We apply the drift theorem (Theorem 5) to a potential that reflects the number of 0-bits in the current window. Consider the interval $[\eta_{\mathrm{lb}}\beta n, (\eta_{\mathrm{lb}} + \eta_{\mathrm{ub}})/2 \cdot \beta n]$ and observe that by assumption the algorithm starts with a potential of at least $(\eta_{\mathrm{lb}} + \eta_{\mathrm{ub}})/2 \cdot \beta n$. Using Lemma 7 with the condition from the first paragraph and Lemma 6, if the current potential is within the interval and the end of the path is not reached, then the expected increase in the potential is bounded from below by a positive constant.

For $j \in \mathbb{N}_0$ the probability that the potential decreases by j is bounded from above by the probability that the (1+1) EA flips at least j bits. This probability is at most $\binom{n}{j}(c/n)^j \leq c^j/(j!) \leq (ec/j)^j \leq 2^{-j} \cdot 2^{2ec}$, where the last estimation is trivial for $j \leq 2ec$ and obvious otherwise. Applying Theorem 5 with $\delta = 1$ and $r = 2^{2ec}$ yields that with overwhelming probability in $2^{\kappa n}$ generations, if again κ is sufficiently small, the potential does not decrease below $\eta_{\mathrm{lb}}\beta n$ or the end of the path is reached. □

5.3 Hitting the Path

All that is left to complete the proof of the main result is the fact that the path is reached from a random initialization, with overwhelming probability.

Our function is constructed such that after a typical initialization the fitness equals OneMax on all bits outside the window B_1, multiplied by a huge weight of 2^n, plus BinVal on all bits inside the window. The OneMax-part encourages the (1+1) EA to turn the bits outside the window B_1 to 1 quickly. Note that B_1 becomes a potential window once the number of 0-bits outside B_1 is no more than αn. The BinVal-part on the bits within B_1 is used to maintain a certain number of 0-bits inside the window. This ensures that the algorithm reaches the path close to B_1, for the same reason that prevents the (1+1) EA from taking shortcuts when climbing the path. This reasoning is made precise in the following lemma.

LEMMA 10: *Let $0 < \alpha < \beta < \gamma < 1$, $0 < \eta_{\mathrm{lb}} < \eta_{\mathrm{ub}} < 1/2$, and $c \geq 16$ be constants such that $\eta_{\mathrm{lb}} - \alpha/\beta > \gamma$ and $\beta/(1-\beta) < \gamma < 2\beta/(1-\beta)$, $c\beta > \frac{2-2\eta_{\mathrm{ub}}}{1-2\eta_{\mathrm{ub}}}$, and $\alpha < \frac{1-2\eta_{\mathrm{ub}}}{3c}$. Let f_Π, with respect to α, β, and γ, be constructed as in Definition 2, for Π chosen uniformly at random. With probability $1 - 2^{-\Omega(n)}$ the (1+1) EA with mutation probability $p(n) = c/n$ optimizing f_Π at some point of time reaches some search point x with $i_x^* \leq \beta n$ and $|x_{|B_{i_x^*}}|_0 \geq (\eta_{\mathrm{lb}} + \eta_{\mathrm{ub}})/2 \cdot \beta n$.*

PROOF: The proof follows from reusing many previous arguments, as the situation of the (1+1) EA moving toward the path is very similar to climbing up the path. The outline of the proof is as follows. We first show that a minimum number of 0-bits in B_1—the same minimum number as in the setting of climbing the path—prevents the (1+1) EA from taking shortcuts. We then show that the path is reached within the first n^2 generations. Finally, we argue that, during these n^2 generations, we keep a minimum number of 0-bits inside the window, with overwhelming probability.

Let x be the current search point of the (1+1) EA. By the same reasoning as in Lemma 5, we observe that if $|x_{|B_1}|_0 \geq \eta_{\mathrm{lb}}\beta n$, then for every $j > \beta n$, since $\eta_{\mathrm{lb}}\beta n > \alpha n + \gamma\beta n$ and $|B_1 \cap B_j| \leq \gamma\beta n$, we have $j \notin \mathcal{B}_x$. Hence, we only need to prove that the number of 0-bits in B_1 does not decrease below $\eta_{\mathrm{lb}}\beta n$ until the set $\mathcal{B}_x$ of potential window positions becomes nonempty for the first time.

The set $\mathcal{B}_x$ is nonempty if the number of 0-bits outside of B_1 has decreased toward a value of at most αn. Every mutation decreasing the number of 0-bits outside of B_1 is accepted. Such a mutation has a probability of at least

$$\alpha n \cdot \frac{c}{n}\left(1 - \frac{c}{n}\right)^{n-1} = \Omega(1).$$

Hence, there is a constant $\kappa > 0$ such that for any initialization, the expected number of generations until the number of 0-bits has decreased to a value of at most αn is at most κn. By Markov's inequality, the probability that this has not happened after $2\kappa n$ generations is at most 1/2. The probability that this still has not happened after $\lfloor n^2/(2\kappa n)\rfloor = \Omega(n)$ periods, each of $2\kappa n$ generations, is $2^{-\Omega(n)}$. Hence, with probability $1 - 2^{-\Omega(n)}$ the path is found within n^2 generations. In the following we assume that this happens.

Recall that we have $|B_i| = \ell = \beta n$ and $\eta_{\text{ub}} < 1/2$, constant. The initial search point contains an expected number of $1/2 \cdot \beta n$ 0-bits in B_1. The probability that the initial search point contains at least $\eta_{\text{ub}}\beta n$ 0-bits in B_1 is $1 - 2^{-\Omega(n)}$ by Chernoff bounds. Assume that this happens and consider a situation where we have at least αn 0-bits outside of B_1 and the number of 0-bits in B_1 has decreased below $\eta_{\text{ub}}\beta n$. Arguing as in the proof of Lemma 7, if the number of 0-bits in B_1 is within $\eta_{\text{lb}}\beta n$ and $\eta_{\text{ub}}\beta n$, then there is a positive drift toward increasing the number of 0-bits again. (The only difference from the previous arguments is as follows. Instead of considering a new random BinVal instance when the current i^*_x-value is increased, we obtain a new BinVal instance whenever the number of 1-bits outside the window is increased. The probability for the latter event can even be larger than the probability for the former.) This allows us to apply Lemma 8 in the same fashion as in the proof of Lemma 7. This results in a positive drift. Since we start with at least $\eta_{\text{ub}}\beta n$ 0-bits in B_1, we can apply the drift theorem as in Lemma 18 w.r.t. the interval $[(\eta_{\text{lb}} + 2\eta_{\text{ub}})/3 \cdot \beta n, \eta_{\text{ub}}\beta n]$. This proves that in n^2 generations the number of 0-bits in B_1 does not drop to or below $(\eta_{\text{lb}} + 2\eta_{\text{ub}})/3 \cdot \beta n$, with probability $1 - 2^{-\Omega(n)}$.

We only have to deal with one further caveat. If x^* is the first point on the path, then the above arguments on the 0-bits inside B_1 do not apply for the generation in which x^* is created. This is because every point on the path is better than every point y with $\mathcal{B}_y = \emptyset$, and so selection works differently in this special generation. We resort to a more direct argument to prove that not many 0-bits are lost. Consider the mutation that creates x^*. Since x^* is the first search point where $\mathcal{B}_{x^*} \neq \emptyset$, its parent must have had more than αn 0-bits outside of B_1. It also had at least $(\eta_{\text{lb}} + 2\eta_{\text{ub}})/3 \cdot \beta n$ 0-bits inside B_1. The probability that during mutation more than $(\eta_{\text{ub}} - \eta_{\text{lb}})/6 \cdot \beta n$ bits were flipped is $n^{-\Omega(n)}$. Hence with overwhelming probability the number of 0-bits in x^* is still at least

$$\begin{aligned}\left|x^*\right|_0 &\geq \alpha n + (\eta_{\text{lb}} + 2\eta_{\text{ub}})/3 \cdot \beta n - (\eta_{\text{ub}} - \eta_{\text{lb}})/6 \cdot \beta n\\ &\geq \alpha n + (\eta_{\text{lb}} + \eta_{\text{ub}})/2 \cdot \beta n.\end{aligned}$$

Along with Lemma 3, this yields that $|x^*_{|B_{i^*_x}}|_0 = |x|_0 - \alpha n \geq (\eta_{\text{lb}} + \eta_{\text{ub}})/2 \cdot \beta n$ as claimed. As the sum of all error probabilities is $2^{-\Omega(n)}$, the claim follows. □

5.4 Putting Everything Together

Now we are prepared to prove Theorem 4.

Proof of Theorem 4: Choose $\eta_{\text{lb}} := \frac{30}{112}$ and $\eta_{\text{ub}} := \frac{30}{111}$. It is easily verified that for the chosen values $0 < \alpha < \beta < \gamma < 1$, $c\beta > \frac{2-2\eta_{\text{ub}}}{1-2\eta_{\text{ub}}}$, $\eta_{\text{lb}} - \alpha/\beta > \gamma$, $\beta/(1-\beta) < \gamma < 2\beta/(1-\beta)$, and $\alpha < \frac{1-2\eta_{\text{ub}}}{3c}$ holds, satisfying all preconditions on these variables for Lemmas 5, 9, and 10. By Lemma 10, the (1+1) EA reaches some search point x with $i^*_x \leq \beta n$ and $|x_{|B_{i^*_x}}|_0 \geq (\eta_{\text{lb}} + \eta_{\text{ub}})/2 \cdot \beta n$ with overwhelming probability. Lemma 9 then states that with probability $1 - 2^{-\Omega(n)}$, the number of 0-bits in the current window is always at least $\eta_{\text{lb}}\beta n$ until the end of the path is reached or $2^{\kappa n}$ generations have passed for a sufficiently small constant $\kappa > 0$ (which would correspond to the claimed time bound).

Given the condition on the 0-bits, by Lemma 5 the (1+1) EA increases its current i^*_x-value by at most βn in one generation, with probability $1 - n^{-\Omega(n)}$. The probability that this always happens until an i^*_x-value of L' is reached is at least $1 - L' \cdot n^{-\Omega(n)} = 1 - n^{-\Omega(n)}$ since $L' = 2^{\Theta(n)}$. This implies that (1+1) EA spends at least $L'/(\beta n) - 1 \geq 2^{\kappa n}$ generations on the path, with probability $1 - 2^{-\Omega(n)}$, if κ is chosen small enough. Since the sum of all error probabilities is $2^{-\Omega(n)}$, the claim follows. □

6 Conclusions

Understanding which problems and problem classes are difficult for evolutionary algorithms remains a challenging task. We have made an important step forward by showing that even innocent looking functions like monotonic ones can be surprisingly hard to optimize with evolutionary algorithms. We showed that the optimum of any monotonic function is found efficiently if the mutation probability is at most $1/n$. Once the mutation probability exceeds $16/n$, the situation drastically changes. In this case, there are monotonic functions such that the (1+1) EA with overwhelming probability needs an exponential time to find their optimum.

This result indicates that, to a greater extent than expected, care has to be taken when choosing the mutation probability, even if restricting oneself to mutation probabilities c/n with a constant c. Contrary to previous observations, for example, for linear functions, it may well happen that constant factor changes in the mutation probability lead to more than constant factor changes in the efficiency.

Mutation probabilities of $16/n$ are not used in practice in evolutionary algorithms. However, in memetic algorithms and artificial immune systems they are actually applied. Therefore, our theoretical findings make a significant contribution toward practical applications of these randomized search heuristics.

Apart from generally suggesting more research on the right mutation probability, this work leaves two particular problems open. (1) For the mutation probability $1/n$, give a sharp upper bound for the optimization time of monotonic functions (this order of magnitude is between $\Omega(n \log n)$ and $O(n^{3/2})$). (2) Determine the largest constant c such that the expected optimization time of the (1+1) EA with mutation probability $p(n) = c/n$ is $n^{O(1)}$ on every monotonic function. Currently, we only know that $1 < c < 16$ holds. We do not expect the pessimistic model that establishes the $O(n^{3/2})$ bound for $c = 1$ (Jansen, 2007) to be particularly useful for this task. In this model it is harder to locate the unique optimum than for any monotonic function. Note that it is not even clear that the upper bound is tight for $c = 1$ on monotonic functions.

Acknowledgments

The authors would like to thank Xin Yao for several useful discussions. The first author is thankful to Jon Rowe for pointing out this problem to him at the ThRaSH workshop in Birmingham. This material is based in part upon works supported by the Science Foundation Ireland under Grant No. 07/SK/I1205. Dirk Sudholt was partly supported by a postdoctoral fellowship from the German Academic Exchange Service while visiting the International Computer Science Institute in Berkeley, CA, USA and EPSRC grant EP/D052785/1. Carola Winzen is a recipient of the Google Europe Fellowship in Randomized Algorithms. This research is supported in part by this Google Fellowship. Christine Zarges was partly supported by a postdoctoral fellowship from the German Academic Exchange Service.

References

Aldous, D. (1983). Minimization algorithms and random walk on the d-cube. *Annals of Probability*, 11:403–413.

Alon, N., and Spencer, J. H. (2008). *The probabilistic method* (3rd ed.). New York: Wiley.

Bäck, T., Fogel, D., and Michalewicz, Z. (Eds.) (1997). *Handbook of evolutionary computation*. Oxford, UK: Oxford University Press.

Böttcher, S., Doerr, B., and Neumann, F. (2010). Optimal fixed and adaptive mutation rates for the leading ones problem. In *11th International Conference on Parallel Problem Solving from Nature (PPSN XI). Lecture Notes in Computer Science, Part I*, Vol. 6238 (pp. 1–10). Berlin: Springer.

Cervantes, J., and Stephens, C. R. (2009). Limitations of existing mutation rate heuristics and how a rank GA overcomes them. *IEEE Transactions on Evolutionary Computation*, 13:369–397.

Cormen, T. H., Leiserson, C. E., Rivest, R. L., and Stein, C. (2001). *Introduction to algorithms* (2nd ed.). Cambridge, MA: MIT Press.

Dasgupta, D., and Niño, L. F. (2008). *Immunological computation: Theory and applications*. Boston: Auerbach.

Doerr, B. (2011). Analyzing randomized search heuristics: Tools from probability theory. In A. Auger and B. Doerr (Eds.), *Theory of randomized search heuristics*, Vol. 1 of *Series on Theoretical Computer Science* (pp. 1–20). Singapore, World Scientific.

Doerr, B., and Goldberg, L. (2010a). Adaptive drift analysis. In *Proceedings of Parallel Problem Solving from Nature (PPSN XI), Part I, Lecture Notes in Computer Science,* Vol. 6238 (pp. 32–41). Berlin: Springer.

Doerr, B., and Goldberg, L. (2010b). Drift analysis with tail bounds. In *Proceedings of Parallel Problem Solving from Nature (PPSN XI), Part I, Lecture Notes in Computer Science*, Vol. 6238 (pp. 174–183). Berlin: Springer.

Doerr, B., Happ, E., and Klein, C. (in press). Crossover can provably be useful in evolutionary computation. *Theoretical Computer Science*, to appear. doi: DOI: 10.1016/j.tcs.2010.10.035.

Doerr, B., Jansen, T., and Klein, C. (2008). Comparing global and local mutations on bit strings. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2008)*, pp. 929–936.

Doerr, B., Jansen, T., Sudholt, D., Winzen, C., and Zarges, C. (2010). Optimizing monotonic functions can be difficult. In *11th International Conference on Parallel Problem Solving from Nature (PPSN XI), Part I, Lecture Notes in Computer Science,* Vol. 6238 (pp. 42–51). Berlin: Springer.

Doerr, B., Johannsen, D., and Winzen, C. (2010a). Drift analysis and linear functions revisited. In *Proceedings of IEEE Congress on Evolutionary Computation (CEC 2010)*, pp. 1967–1974.

Doerr, B., Johannsen, D., and Winzen, C. (2010b). Multiplicative drift analysis. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2010)*, pp. 1449–1456.

Droste, S., Jansen, T., and Wegener, I. (1998). On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In *Proceedings of Parallel Problem Solving from Nature (PPSN V), Lecture Notes in Computer Science,* Vol. 1498 (pp. 13–22). Berlin: Springer.

Droste, S., Jansen, T., and Wegener, I. (2002). On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276:51–81.

He, J., and Yao, X. (2001). Drift analysis and average time complexity of evolutionary algorithms. *Artificial Intelligence*, 127:57–85.

He, J., and Yao, X. (2002). Erratum to He and Yao (2001). *Artificial Intelligence*, 140:245–248.

Horn, J., Goldberg, D., and Deb, K. (1994). Long path problems. In *Proceedings of Parallel Problem Solving from Nature (PPSN IV), Lecture Notes in Computer Science*, Vol. 866 (pp. 149–158). Berlin: Springer.

Igel, C., and Toussaint, M. (2004). A no-free-lunch theorem for non-uniform distributions of target functions. *Journal of Mathematical Modelling and Algorithms*, 3:313–322.

Jägersküpper, J. (2011). Combining Markov-chain analysis and drift analysis—The (1+1) Evolutionary Algorithm on linear functions reloaded. *Algorithmica*, 59(3):409–424.

Jansen, T. (2007). On the brittleness of evolutionary algorithms. In *Proceedings of Foundations of Genetic Algorithms (FOGA 2007), Lecture Notes in Computer Science,* Vol. 4436 (pp. 54–69). Berlin: Springer.

Jansen, T., Oliveto, P. S., and Zarges, C. (2011). On the analysis of the immune-inspired b-cell algorithm for the vertex cover problem. In *Proceedings of the International Conference on Artificial Immune Systems (ICARIS 2011)*, pp. 117–131.

Jansen, T., and Wegener, I. (2000). On the choice of the mutation probability for the (1+1) EA. In *Proceedings of Parallel Problem Solving from Nature (PPSN VI), Lecture Notes in Computer Science,* Vol. 1917 (pp. 89–98). Berlin: Springer.

Jansen, T., and Zarges, C. (2011). Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theoretical Computer Science*, 412:517–533.

Kelsey, J., and Timmis, J. (2003). Immune inspired somatic contiguous hypermutations for function optimisation. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2003)*, pp. 207–218.

Krasnogor, N., and Smith, J. (2005). A tutorial for competent memetic algorithms: Model, taxonomy, and design issues. *IEEE Transactions on Evolutionary Computation*, 9(5):474–488.

Mitzenmacher, M., and Upfal, E. (2005). *Probability and computing: Randomized algorithms and probabilistic analysis*. Cambridge, MA: Cambridge University Press.

Mühlenbein, H. (1992). How genetic algorithms really work. Mutation and hillclimbing. In *Proceedings of Parallel Problem Solving from Nature (PPSN II), Lecture Notes in Computer Science*, Vol. 6825 (pp. 15–25). Berlin: Springer.

Ochoa, G. (2002). Setting the mutation rate: Scope and limitations of the $1/L$ heuristic. In *Proceedings of Genetic and Evolutionary Computation Conference (GECCO 2002)*, pp. 495–502.

Oliveto, P., He, J., and Yao, X. (2007). Time complexity of evolutionary algorithms for combinatorial optimization: A decade of results. *International Journal of Automation and Computing*, 4:281–293.

Oliveto, P. S., and Witt, C. (2011). Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica*, 59:369–386.

Sudholt, D. (2011). A new method for lower bounds on the running time of evolutionary algorithms. *ArXiv e-prints*. Available from http://arxiv.org/abs/1109.1504

Witt, C. (2011). Tight bounds on the optimization time of the (1+1) EA on linear functions. *ArXiv e-prints*. Available from http://arxiv.org/abs/1108.4386v1