Algorithmica (2018) 80:1732–1768
https://doi.org/10.1007/s00453-017-0341-1

Static and Self-Adjusting Mutation Strengths for Multi-valued Decision Variables

Benjamin Doerr[1] · **Carola Doerr**[2] · **Timo Kötzing**[3]

Received: 18 October 2016 / Accepted: 26 June 2017 / Published online: 5 July 2017

Abstract The most common representation in evolutionary computation are bit strings. With very little theoretical work existing on how to use evolutionary algorithms for decision variables taking more than two values, we study the run time of simple evolutionary algorithms on some OneMax-like functions defined over $\Omega = \{0, 1, \ldots, r-1\}^n$. We observe a crucial difference in how we extend the one-bit-flip and standard-bit mutation operators to the multi-valued domain. While it is natural to modify a random position of the string or select each position of the solution vector for modification independently with probability $1/n$, there are various ways to then change such a position. If we change each selected position to a random value different from the original one, we obtain an expected run time of $\Theta(nr \log n)$. If we change each selected position by $+1$ or -1 (random choice), the optimization time reduces to $\Theta(nr + n \log n)$. If we use a random mutation strength $i \in \{0, 1, \ldots, r-1\}$ with probability inversely proportional to i and change the selected position by $+i$ or $-i$ (random choice), then the optimization time becomes $\Theta(n \log(r)(\log n + \log r))$, which is asymptotically faster than the previous if $r = \omega(\log(n) \log\log(n))$. Interestingly, a better expected performance can be achieved with a *self-adjusting mutation strength* that is based on the success of previous iterations. For the mutation operator that modifies a randomly chosen position, we show that the self-adjusting mutation

Results presented in this work are based on [12,13].

✉ Timo Kötzing
Timo.Koetzing@hpi.de

1 LIX - UMR 7161, École Polytechnique, CS35003, 91120 Palaiseau, France

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

3 Hasso-Plattner-Institut, Prof.-Dr.-Helmert-Str. 2-3, 14482 Potsdam, Germany

strength yields an expected optimization time of $\Theta(n(\log n + \log r))$, which is best possible among all dynamic mutation strengths. In our proofs, we use a new multiplicative drift theorem for computing lower bounds, which is not restricted to processes that move only towards the target.

Keywords Theory of randomized search heuristics · Runtime analysis · Genetic algorithms · Parameter choice · Parameter control

1 Introduction

In evolutionary computation, taking ideas both from computer science and biology, often search and optimization problems are modeled in a way that the solution candidates are fixed-length strings over the alphabet consisting of 0 and 1. In other words, the search space Ω is chosen to be $\{0, 1\}^n$ for some positive integer n. Such a representation of solution candidates is very suitable to model binary decision variables. For example, when searching for graph substructures like large cliques, (degree-constrained) spanning trees, or certain matchings, we can use binary decision variables describing whether a vertex or an edge is part of the solution or not. For these reasons, the bit string representation is by far the most prominent one in evolutionary computation.

When a problem intrinsically consists of other types of decision variables, the algorithm designer has the choice to either work with a different representation (e.g., permutations in the traveling salesman problem, see also [43] for other early examples of experimental results investigating discrete search spaces different from $\{0, 1\}^n$) or to reformulate the problem using a bit string representation. For an example for the latter, see, e.g., [19], where the Eulerian cycle problem (asking for a permutation of the edges) was re-modeled as a matching problem in the adjacency lists. While the re-modeling in [19] led to a significantly improved runtime, in general, there is the risk that such a re-modeling leads to a less natural and less efficient optimization process. In such cases, it may be superior to work with a representation different from bit strings. The traveling salesman problem is an example for a problem where a convincing bit string representation has not been found so far.

While in this work we shall not deal with the difficulties of treating permutation search spaces in evolutionary computation, we shall try to extend our good understanding of the bit string representation to representations in which the decision variables can take more values than just zero and one. Consequently, we shall work with search spaces $\Omega = \{0, \dots, r-1\}^n$. Such search spaces are a natural representation when each decision variable can take one out of r values. Examples from the evolutionary computation literature include scheduling n jobs on r machines, which naturally leads to the search space $\{0, \dots, r-1\}^n$, see Gunia [26]. However, also rooted trees lead to this type of representation: Since each vertex different from the root has a unique predecessor in the tree, a rooted tree on n vertices can be represented via an element of $\{0, \dots, n-1\}^{n-1}$. This was exploited in [44] to design evolutionary algorithms for shortest-path problems.

Our focus in this work is to analyze how to best deal with such search spaces in a direct manner, that is, without remodeling the problem into a bit string representation.

Our motivation is that this seems to us the most natural approach. It is clear that a variable taking values in $\{0,\dots,r-1\}$ can be encoded in $\log r$ bits, which immediately gives a reformulation of the problem in the language of bit strings. We discuss briefly in Sect. 2.1 such encodings. However, by using a binary encoding and then standard mutation operators, we end up with algorithms ignoring the structure of the search space. For this reason, in this work, we prefer to investigate how the existing work on binary representations can be extended directly to multi-valued decision variables.

1.1 Static and Self-Adjusting Mutation Operators for Multi-valued Search Spaces

A first question, and our main focus in this work, is what mutation operators to use in such multi-valued search spaces. When there is no particular topology in the components $i \in [n] := \{1,\dots,n\}$, that is, in each dimension with values from $[0\dots r-1] := \{0\} \cup [r-1]$, then the natural analogue of the standard-bit mutation operator is to select each component $i \in [n]$ independently and mutate the selected components by changing the current value to a random other value in $[0\dots r-1]$. This operator was used in [26,44] as well as in the theoretical works [20,22].

When the decision values $0,1,\dots,r-1$ carry more meaning than just denoting alternatives without particular topology, then one may want to respect this in the mutation operator. We shall not discuss the most general set-up of a general distance matrix defined on the values $0,1,\dots,r-1$, but assume that they represent linearly ordered alternatives.

Given such a linear topology, several other mutation operators suggest itself. We shall always imitate either the one-bit-flip mutation operator (changing a random position) or the standard-bit mutation operator (changing each component $i \in [n]$ independently with probability $1/n$). Hence the only point of discussion is how a component selected for modification is then changed. Since this is merely the question by how far the existing entry is moved in the range $\{0,\dots,r-1\}$, we call this to be determined information the *mutation strength*. We also call the change of a single entry of the solution vector an *elementary mutation*.

The principle that mutation is a minimalistic change of the individual suggests to alter a selected component randomly by $+1$ or -1 (for a precise definition, including also a description of how to treat the boundary cases, see again Sect. 2). We say that this mutation operator has a *mutation strength* equal to one. Naturally, a mutation strength of one carries the risk of being slow—it takes $r-1$ such elementary mutations to move one component from one boundary value, say 0, to the other, say $r-1$.

In this language, the previously discussed mutation operator changing a selected component to a new value chosen uniformly at random can (roughly) be described as having a mutation strength chosen uniformly at random from $[r-1]$. While this operator does not have the disadvantage of moving slowly through the search space, it does have the weakness that reaching a particular target is slow, even when already close to it.

Based on these (intuitive, but we shall make them precise later) observations, we propose an elementary mutation that takes a biased random choice of the mutation strength. We give more weight to small steps than the uniform operator, but do allow larger jumps with certain probability. More precisely, in each elementary mutation independently we choose the mutation strength randomly such that a jump of $+j$ or $-j$ occurs with probability inversely proportional to j, and hence with probability $\Theta((j \log r)^{-1})$. This distribution, called *harmonic distribution*, was used in [7] to design fast greedy random walks in the one-dimensional domain $\{0, \ldots, r-1\}$.

All three mutation operators described above are *static*, i.e., the mutation strength (more precisely, its distribution) does not change throughout the run of the algorithms. For many practical problems (and few theoretical scenarios), however, *adaptive parameter choices* are known to outperform such static ones, cf. the discussion in Sect. 7.1.

Our analyses of the three different static parameter choices suggest that also for the multi-valued OneMax problems considered in this work an adaptive parameter choice could be useful. Indeed, as briefly mentioned above we shall observe that the ± 1 mutation strength is too slow in the beginning of the optimization process while the uniform mutation strength often suggests too large mutation strengths towards the end of the optimization. The harmonic operator seems more balanced in the choice of the mutation strength, which, however, still means that it does not prefer the currently ideal mutation strength, but has to wait until its random choice selects a suitable mutation strength.

These observations inspire us to invent a self-adjusting choice of the mutation strength. Each position $i \in [n]$ is equipped with a *step size* $v_i \in [r/4]$. If the i-th position is selected for variation, then with probability $1/2$ the current value x_i is increased to $x_i + v_i$ and it is decreased to $x_i - v_i$ otherwise (all technical details, such as the treatment of the boundary cases will be discussed in Sect. 2). Has the iteration been successful, that is, if at the end of the iteration the fitness of the offspring is strictly better than that of its parent, then for all positions i changed in the mutation step, the step size v_i is increased to av_i (for some constant $a > 1$). If the new search point has worse fitness, it is discarded and the step size v_i of all positions i involved is decreased to bv_i (for a positive constant $b < 1$).

We shall see that this self-adjusting choice together with one-bit-mutations (changing one randomly chosen position) outperforms the three static mutation strengths both for one-bit-mutations and standard-bit mutation. In fact, we obtain a performance that is optimal among a broad class of black-box optimizers. We conjecture that the same asymptotic performance is obtained when using the self-adjusting mutation strength with standard-bit mutation, but we have to leave this an open problem.

1.2 Run Time Analysis of Multi-valued OneMax Functions

To gain a rigorous understanding of the working principles of the different mutations strengths, we conduct a mathematical run time analysis for simple evolutionary algorithms on multi-valued analogues of the OneMax test function. Comparable approaches have been very successful in the past in studying in isolation particular aspects of evolutionary computation, see, e.g., [31]. Also, many observations first made in such simplistic settings have later been confirmed for more complicated algorithms (see, e.g., [1]) or combinatorial optimization problems (see, e.g., [40]).

On bit strings, the classic ONEMAX test function is defined by $\text{OM} : \{0,1\}^n \to [0 \dots n]; (x_1, \dots, x_n) \mapsto \sum_{i=1}^n x_i$. Due to the obvious symmetry, for most evolutionary algorithms it makes no difference whether the target is to maximize or to minimize this function. For several reasons, among them the use of drift analysis, in this work it will be more convenient to always assume that our target is the *minimization* of the given objective function.

The obvious multi-valued analogue of this ONEMAX function is $\text{OM} : [0 \dots r-1]^n \to [0 \dots n(r-1)]; x \mapsto \sum_{i=1}^n x_i$, however, a number of other functions can also be seen as multi-valued analogues. For example, we note that in the bit string setting we have $\text{OM}(x) = H(x, (0, \dots, 0))$, where $H(x, y) := |\{i \in [n] \mid x_i \neq y_i\}|$ denotes the Hamming distance between two bit strings x and y. Defining $f_z : \{0,1\}^n \to [0 \dots n]; x \mapsto H(x, z)$ for all $z \in \{0,1\}^n$, we obtain a set of 2^n objective functions that all have an isomorphic fitness landscape. Taking this route to define multi-valued analogue of ONEMAX functions, we obtain the class of functions $f_z : [0 \dots r-1]^n \mapsto [0 \dots n(r-1)]; x \mapsto \sum_{i=1}^n |x_i - z_i|$ for all $z \in [0 \dots r-1]^n$, again with $f_{(0,\dots,0)}$ being the ONEMAX function defined earlier. Note that these objective functions do not all have an isomorphic fitness landscape. The asymmetry with respect to the optimum z can be overcome by replacing the classic distance $|x_i - z_i|$ in the reals by the distance modulo r (ring distance), that is, $\min\{x_i - (z_i - r), |x_i - z_i|, (z_i + r) - x_i\}$, creating yet another non-isomorphic fitness landscape. All results we show in the following hold for all these objective functions.

We study the performance of the two elementary black-box optimization algorithms *randomized local search (RLS)* and the *(1+1) evolutionary algorithm (EA)* on these test functions. RLS and the (1+1) EA are arguably among the most simple randomized search heuristics, but many results that could first only be shown for (one or both of) these two algorithms could later be extended to more complicated algorithms, making them an ideal instrument for a first study of a new subject. For the different ways of setting the mutation strength, we conduct a mathematical run time analysis, that is, we prove bounds on the expected number of iterations the algorithms need to find an optimal solution. This *optimization time* today is one of the most accepted performance measures for evolutionary algorithms.

1.3 Our Results

As mentioned, we analyze the performance of different extensions of RLS and the (1 + 1) EA on the r-valued ONEMAX functions described above. All variants maintain the property that for RLS in each iteration the entry of exactly one position $i \in [n]$ is changed, while for the (1 + 1) EA for each $i \in [n]$ an independent coin flip with success probability $1/n$ decides whether or not the entry of the i-th position is subject to change. In both algorithms, we study how different ways to change entries selected for modification influence the run time. For the three static mutation strength we analyze the performance of the respective RLS and (1 + 1) EA variant; for the self-adjusting

Table 1 Expected run times on r-valued ONEMAX functions for (a) the (1 + 1) EA and RLS using different static mutation strengths (first three rows of results); and for (b) RLS with self-adjusting mutation strength

Mutation strength	Expected run time	Scaling w.r.t. r
Uniform	$\Theta(rn \log n)$	$\Theta(r)$
± 1	$\Theta(n \log n + nr)$	$\Theta(r)$
Harmonic	$\Theta(n \log r(\log n + \log r))$	$\Theta((\log r)^2)$
Self-adjusting	$\Theta(n(\log n + \log r))$	$\Theta(\log r)$

variation operator we only regard the extension of RLS and leave the analysis of the corresponding (1 + 1) EA variant as an open problem.

For the uniform mutation strength, we show tight and precise (including the leading constant) run time guarantee of $(1 \pm o(1))(r-1)n\ln(n)$ for RLS and $(1 \pm o(1))e(r-1)n\ln(n)$ for the (1 + 1) EA, which are valid for all values of r (Sect. 4). For the cautious ± 1 mutation strength, the run time improves to $\Theta(n(r + \log n))$ for both RLS and the (1 + 1) EA (Sect. 5). The harmonic mutation strength overcomes the linear influence of r and gives a run time of $\Theta(n \log(r)(\log r + \log n))$, which for most values of r is significantly better than the previous bounds (Sect. 6).

For RLS with self-adjusting mutation strength, we show that for reasonable choices of the update constants a and b, e.g., $a \in [1.6, 2]$ and $b = \sqrt[4]{1/a}$ imitating the 1/5-th rule, the expected optimization time drops to $\Theta(n(\log n + \log r))$, thus gaining a factor of at least $\log r$ over any RLS or (1+1) EA variant using static step sizes (indeed, as we shall discuss in Sect. 6.1, from [7] we know that no static distribution of step sizes can achieve a better run time than $\Omega((\log r)^2)$ for $n = 1$). A simple information-theoretic consideration shows that the lower bound $\Omega(n \log r)$ applies to all comparison-based algorithms, while $\Omega(n \log n)$ is known to be a lower bound for the performance of any unary unbiased black-box algorithm (cf. [37] for a proof of this result and a discussion of unbiased variation). Our results are summarized in Table 1.

All our run time analyses rely on drift methods. For the lower bound for uniform mutation strengths we prove a variant of the multiplicative drift lower bound theorem [45] that does not need the restriction that the process cannot go back to inferior search points (see Sect. 3.1). We expect that this will find other applications in the future.

1.4 Related Works

In particular for the situation that r is large, one might be tempted to think that results from continuous optimization can be helpful. So far, we were not successful in this direction. A main difficulty is that in continuous optimization, usually the asymptotic rate of convergence is regarded. Hence, when operating with a fixed r in our setting and re-scaling things into, say, $\{0, \frac{1}{r}, \frac{2}{r}, \ldots, 1\}^n$, then these results, due to their asymptotic nature, could become less meaningful. For this reason, the only work in the continuous domain that we found slightly resembling ours is by Jägersküpper (see [30] and the references therein), which regards continuous optimization with an a-priori fixed target precision. However, the fact that Jägersküpper regards approximations with respect to the Euclidean norm (in other words, minimization of the sphere function) makes his results hard to compare to ours, which can be seen as minimization of the 1-norm.

Coming back to the discrete domain, as said above, the vast majority of theoretical works on evolutionary computation work with a bit string representation. A notable exception is the work on finding shortest path trees (e.g., [44]); however, in this setting we have that the dimension and the number r of values are not independent: one naturally has r equal to the dimension, because each of the $n-1$ non-root vertices has to choose one of the $n-1$ other vertices as predecessor.

Therefore, we see only three previous works that are comparable to ours. The first two regard the optimization of linear functions via the $(1+1)$ EA using mutation with uniform strength, that is, resetting a component to a random other value. The main result of [20] is that the known run time bound of $O(n \log n)$ on linear functions defined on bit strings remains valid for the search space $\{0, 1, 2\}^n$. This was extended and made more precise in [22], where for r-valued linear functions an upper bound of $(1+o(1))e(r-1)n\ln(n) + O(r^3 n \log\log n)$ was shown together with a $(1+o(1))n(r-1)\ln(n)$ lower bound.

A third paper considers dynamically changing fitness functions [35]. They also consider OneMax functions with distance modulo r, using ± 1 mutation strength. In this setting the fitness function changed over time and the task was to track it as closely as possible, which the ± 1 mutation strength can successfully do. Note that a seemingly similar work on the optimization of a dynamic variant of the maze function over larger alphabets [38] is less comparable to our work since there all non-optimal values of a decision variable contribute the same to the fitness function.

Compared to these works, we only regard the easier static OneMax problem (note though that there are several ways to define multi-valued OneMax functions), but obtain tighter results also for larger values of r and for four different mutation strengths.

As for the self-adjusting mutation operator (related literature will be discussed in Sect. 7), we remark that after the work presented in [10] our result is only the second time that a self-adjusting parameter setting is proven to outperform any static choice for a discrete optimization problem, and it is the first time that this is shown for a problem over multiple decision variables.

2 Algorithms and Problems

In this section we define the algorithms and problems considered in this paper. We let $[r] := \{1, \ldots, r\}$ and $[0 \ldots r] := \{0\} \cup [r]$. For a given search space Ω, a fitness function is a function $f : \Omega \to \mathbb{R}$. While a frequently analyzed search space is $\Omega = \{0, 1\}^n$, we will consider in this paper $\Omega = [0 \ldots r-1]^n$.

We define the following two metrics on $[0 \ldots r-1]$, called *interval-metric* and *ring-metric*, respectively. The intuition is that the interval metric is the usual metric induced by the metric on the natural numbers, while the ring metric connects the two endpoints of the interval (and, thus, forms a ring). Formally we have, for all $a, b \in [0 \ldots r-1]$,

$$d_{\text{int}}(a, b) = |b - a|;$$

$$d_{\text{ring}}(a, b) = \min\{|b - a|, |b - a + r|, |b - a - r|\}.$$

We consider different *step operators* step : $[0 \dots r - 1] \to [0 \dots r - 1]$ (possibly randomized). These step operators will later decide the update of a mutation in a given component. Thus we call, for any given $x \in [0 \dots r - 1]$, $d(x, \text{step}(x))$ the *mutation strength*. We consider the following step operators.

– The *uniform step* operator chooses a different element from $[0 \dots r - 1]$ uniformly at random; thus we speak of a *uniform mutation strength*.
– The ± 1 operator chooses to either add or subtract 1, each with probability $1/2$; this operator has a mutation strength of 1.
– The *harmonic* operator makes a jump of size $j \in [1 \dots r - 1]$ with probability proportional to $1/j$, choosing the direction uniformly at random; we call its mutation strength *harmonic mutation strength*.

Note that, in the case of the ring-metric, all steps are implicitly considered with wrap-around. For the interval-metric, we consider all steps that overstep a boundary of the interval as invalid, the resulting non-individual is considered to have a fitness worse than all regular individuals $x \in \Omega$.

We consider the algorithms RLS and (1 + 1) EA as given by Algorithms 1 and 2. Both algorithms sample an initial search point from $[0 \dots r - 1]^n$ uniformly at random. They then proceed in rounds, each of which consists of a mutation and a selection step. Throughout the whole optimization process the algorithms maintain a population size of one, and the individual in this population is always the most recently sampled best-so-far solution. The two algorithms differ only in the *mutation operation*. While the RLS makes a step in exactly one position (chosen uniformly at random), the (1 + 1) EA makes, in each position, a step with probability $1/n$.

The fitness of the resulting search point y is evaluated and in the *selection step* the parent x is replaced by its offspring y if and only if the fitness of y is at least as good as the one of x. Since we consider minimization problems here, this is the case if $f(y) \leq f(x)$. Since we are interested in expected *run times*, i.e., the expected number of rounds it takes until the algorithm evaluates for the first time a solution of minimal fitness, we do not specify a termination criterion. For the case of $r = 2$, the two algorithms are exactly the classic Algorithms RLS and (1 + 1) EA, for all three given step operators (which then degenerate to the flip operator, which flips the given bit).

Algorithm 1: RLS minimizing a function $f : [0..r - 1]^n \to \mathbb{R}$ with a given step operator step$(\cdot)$.

```
1 Initialization: Sample $x \in [0..r-1]^n$ uniformly at random;
2 Optimization: for $t = 1, 2, 3, \dots$ do
3     Choose $i \in [n]$ uniformly at random;
4     for $j = 1, \dots, n$ do
5         if $j = i$ then $y_j \leftarrow \text{step}(x_j)$ else $y_j \leftarrow x_j$
6     if $f(y) \leq f(x)$ then $x \leftarrow y$
```

Algorithm 2: The (1 + 1) EA minimizing a function $f : [0..r-1]^n \to \mathbb{R}$ with a given step operator step(·).

```
1 Initialization: Sample $x \in [0..r-1]^n$ uniformly at random;
2 Optimization: for $t = 1, 2, 3, \ldots$ do
3     for $i = 1, \ldots, n$ do
4         With probability $1/n$ set $y_i \leftarrow \text{step}(x_i)$ and set $y_i \leftarrow x_i$ otherwise;
5     if $f(y) \leq f(x)$ then $x \leftarrow y$
```

Let d be either the interval- or the ring-metric and let $z \in [0..r-1]^n$. We can define a straightforward generalization of the OneMax fitness function by

$$\sum_{i=1}^{n} d(x_i, z_i).$$

Whenever we refer to an *r-valued* OneMax *function*, we mean any such function. We refer to d as the *metric of the* OneMax *function* and to z as the *target of the* OneMax *function*.

2.1 Alternative Representations

An alternative representation would be to encode each value from $[0 \ldots r-1]$ in $\log r$ bits, leading to a search space of $\{0,1\}^{n \log r}$. Many encodings are possible, for example the binary coding, see [42] for an extensive discussion.

The binary representation has the weakness that search points with similar fitness can be vastly different: the bit representations $10\ldots0$ and $01\ldots1$ code almost the same value, but are complementary (this is called the *Hamming cliff*). This trap-like behavior can lead to a very poor performance on some OneMax functions, where effectively $\log r$ bits might have to be flipped in order to gain any improvement at all, which takes an expected time of $\Omega(n^{\log r})$.

The disadvantages of the binary representation is overcome by practitioners by using other representations for integers, for example Gray codes [42].

3 Drift Analysis

A central tool in many of our proofs is drift analysis, which comprises a number of tools to derive bounds on hitting times from bounds on the expected progress a process makes towards the target. Drift analysis was first used in evolutionary computation by He and Yao [28] and is now, after a large number of subsequent works, probably the most powerful tool in run time analysis. We briefly collect here the tools we use.

We phrase the following results in terms of a given random process on the real numbers. These theorems can be applied to random processes on some other set Ω by employing a potential function $g : \Omega \to \mathbb{R}$ and thus translating the given process to one on the real numbers. We are mostly interested in the time the process (or its potential) needs to reach 0.

Multiplicative drift is the situation that the progress is proportional to the distance from the target. This quite common situation in run time analysis was first framed into a drift theorem, namely the following one, in [21]. A more direct proof of this results that also gives large deviation bounds was later given in [18].

Theorem 1 *(from [21]) Let $X^{(0)}, X^{(1)}, \ldots$ be a random process taking values in $S := \{0\} \cup [s_{\min}, \infty) \subseteq \mathbb{R}$. Assume that $X^{(0)} = s_0$ with probability one. Assume that there is a $\delta > 0$ such that for all $t \geq 0$ and all $s \in S$ with $\Pr[X^{(t)} = s] > 0$ we have*

$$E[X^{(t+1)}|X^{(t)} = s] \leq (1-\delta)s.$$

Then $T := \min\{t \geq 0 \mid X^{(t)} = 0\}$ satisfies

$$E[T] \leq \frac{\ln(s_0/s_{\min}) + 1}{\delta}.$$

It is easy to see that the upper bound above cannot immediately be matched with a lower bound of similar order of magnitude. Hence it is no surprise that the only lower bound result for multiplicative drift, the following theorem by Witt [45], needs two additional assumptions, namely that the process does not move away from the target and that it does not too often make large jumps towards the target. We shall see later (Theorem 4) that the first restriction can be removed under not too strong additional assumptions.

Theorem 2 *(from [45]) Let $X^{(t)}, t = 0, 1, \ldots$ be random variables taking values in some finite set S of positive numbers with $\min(S) = 1$. Let $X^{(0)} = s_0$ with probability one. Assume that, for all $t \geq 0$,*

$$\Pr[X^{(t+1)} \leq X^{(t)}] = 1.$$

Let $s_{\text{aim}} \geq 1$. Let $0 < \beta, \delta \leq 1$ be such that for all $s > s_{\text{aim}}$ and all $t \geq 0$ with $\Pr[X^{(t)} = s] > 0$, we have

$$E[X^{(t)} - X^{(t+1)} \mid X^{(t)} = s] \leq \delta s,$$
$$\Pr[X^{(t)} - X^{(t+1)} \geq \beta s \mid X^{(t)} = s] \leq \frac{\beta\delta}{\ln(s)}.$$

Then $T := \min\{t \geq 0 \mid X^{(t)} \leq s_{\text{aim}}\}$ satisfies

$$E[T] \geq \frac{\ln(s_0) - \ln(s_{\text{aim}})}{\delta} \frac{1-\beta}{1+\beta}.$$

In situations in which the progress is not proportional to the distance, but only monotonically increasing with it, the following *variable drift* theorem of Johannsen [33] can lead to very good results. Another version of a variable drift theorem can be found in [39, Lemma 8.2].

Theorem 3 *(from [33]) Let $X^{(t)}, t = 0, 1, \ldots$ be random variables taking values in some finite set S of non-negative numbers. Assume $0 \in S$ and let $x_{\min} := \min(S \setminus \{0\})$. Let $X^{(0)} = s_0$ with probability one. Let $T := \min\{t \geq 0 \mid X^{(t)} = 0\}$. Suppose that there exists a continuous and monotonically increasing function $h : [x_{\min}, s_0] \to \mathbb{R}_{>0}$ such that $E[X^{(t)} - X^{(t+1)} | X^{(t)}] \geq h(X^{(t)})$ holds for all $t < T$. Then*

$$E[T] \leq \frac{x_{\min}}{h(x_{\min})} + \int_{x_{\min}}^{s_0} \frac{1}{h(x)} dx.$$

3.1 A New Drift Theorem

In this section, we write $(q)_+ := \max\{q, 0\}$ for any $q \in \mathbb{R}$.

With the drift theorem we prove in this section we want to overcome one difficulty, namely that the only known lower bound theorem for multiplicative drift (Theorem 2) requires that the process does not move away from the target, in other words, that the g-value is non-increasing with probability one. As discussed above, we do not have this property when using the Hamming distance as potential in a run of the $(1+1)$ EA. We solve this problem by deriving from Theorem 2 a drift theorem (Theorem 4 below) that gives lower bounds also for processes that may move away from the optimum. Compared to Theorem 2, we need the stronger assumptions (i) that we have a Markov process and (ii) that we have bounds not only for the drift $g(X^{(t)}) - g(X^{(t+1)})$ or the positive part $(g(X^{(t)}) - g(X^{(t+1)}))_+$ of it, but also for the positive progress $(s - g^{(t+1)})_+$ with respect to any reference point $s \leq g(X^{(t)})$. This latter condition is very natural. In simple words, it just means that we cannot profit from going back to a worse (in terms of the potential) state of the Markov chain.

A second advantage of these stronger conditions (besides allowing the analysis of non-decreasing processes) is that we can easily ignore an initial segment of the process (see Corollary 5). This is helpful when we encounter a larger drift in the early stages of the process. This phenomenon is often observed, e.g., in Lemma 6.7 of [45]. Previous works, e.g., [45], solved the problem of a larger drift in the early stage of the process by manually cutting off this phase. This requires again a decreasing process (or conditioning on not returning to the region that has been cut off) and an extra argument of the type that the process with high probability reaches a search point with potential in $[\tilde{s}_0, 2\tilde{s}_0]$ for a suitable $\tilde{s}_0$. So it is safe to say that Corollary 5 is a convenient way to overcome these difficulties.

Theorem 4 *(multiplicative drift, lower bound, non-decreasing process) Let $X^{(t)}, t = 0, 1, \ldots$ be a Markov process taking values in some set Ω. Let $S \subset \mathbb{R}$ be a finite set of positive numbers with $\min(S) = 1$. Let $g : \Omega \to S$. Let $g(X^{(0)}) = s_0$ with probability one. Let $s_{\text{aim}} \geq 1$. Let*

$$T := \min\{t \geq 0 \mid g(X^{(t)}) \leq s_{\text{aim}}\}$$

be the random variable describing the first point in time for which $g(X^{(t)}) \leq s_{\text{aim}}$.

Let $0 < \beta, \delta \le 1$ *be such that for all* $\omega \in \Omega$*, all* $s_{\text{aim}} < s \le g(\omega)$*, and all* $t \ge 0$ *with* $\Pr[X^{(t)} = \omega] > 0$*, we have*

$$E\left[(s - g(X^{(t+1)}))_+ \mid X^{(t)} = \omega\right] \le \delta s,$$
$$\Pr\left[s - g(X^{(t+1)}) \ge \beta s \mid X^{(t)} = \omega\right] \le \frac{\beta\delta}{\ln(s)}.$$

Then

$$E[T] \ge \frac{\ln(s_0) - \ln(s_{\text{aim}})}{\delta} \frac{1-\beta}{1+\beta} \ge \frac{\ln(s_0) - \ln(s_{\text{aim}})}{\delta}(1 - 2\beta).$$

The proof follows from an application of Witt's drift theorem (Theorem 2) to the random process $Y^{(t)} := \min\{g(X^{(\tau)}) \mid \tau \in [0 \dots t]\}$.

Proof We define a second random process by $Y^{(t)} := \min\{g(X^{(\tau)}) \mid \tau \in [0 \dots t]\}$. By definition, Y takes values in S and Y is decreasing, that is, we have $Y^{(t+1)} \le Y^{(t)}$ with probability one for all $t \le 0$. Trivially, we have $Y^{(0)} = g(X^{(0)}) = s_0$. Let $T_Y := \min\{t \ge 0 \mid Y^{(t)} \le s_{\text{aim}}\}$ be the first time this new process reaches or goes below s_{aim}. Clearly, $T_Y = T$.

Let β, δ as in the theorem. Let $s_{\text{aim}} < s$ and $t \ge 0$ such that $\Pr[Y^{(t)} = s] > 0$. Observe that when $Y^{(t)} = s$, then $Y^{(t)} - Y^{(t+1)} = s - \min\{s, g(X^{(t+1)})\} = (s - g(X^{(t+1)}))_+$. Let A_s^Y be the event that $Y^{(t)} = s$ and let B_ω^X be the event that $X^{(t)} = \omega$. Using the fact that X is a Markov process, we compute

$$\begin{aligned} E\left[Y^{(t)} - Y^{(t+1)} \mid A_s^Y\right] &= \sum_{\omega: g(\omega) \ge s} \Pr\left[A_s^Y \mid B\right] E\left[(s - g(X^{(t+1)}))_+ \mid A_s^Y, B_\omega^X\right] \\ &= \sum_{\omega: g(\omega) \ge s} \Pr\left[B_\omega^X \mid A_s^Y\right] E\left[(s - g(X^{(t+1)}))_+ \mid B_\omega^X\right] \\ &\le \sum_{\omega: g(\omega) \ge s} \Pr\left[B_\omega^X \mid A_s^Y\right] \delta s = \delta s \end{aligned}$$

and

$$\begin{aligned} &\Pr\left[Y^{(t)} - Y^{(t+1)} \ge \beta s \mid A_s^Y\right] \\ &\quad = \sum_{\omega: g(\omega) \ge s} \Pr\left[B_\omega^X \mid A_s^Y\right] \Pr\left[s - X^{(t+1)} \ge \beta s \mid A_s^Y, B_\omega^X\right]] \\ &\quad = \sum_{\omega: g(\omega) \ge s} \Pr\left[B_\omega^X \mid A_s^Y\right] \Pr\left[s - X^{(t+1)} \ge \beta s \mid B_\omega^X\right]] \\ &\quad \le \sum_{\omega: g(\omega) \ge s} \Pr\left[B_\omega^X \mid A_s^Y\right] \frac{\beta\delta}{\ln(s)} = \frac{\beta\delta}{\ln(s)}. \end{aligned}$$

Consequently, Y satisfies the assumptions of the multiplicative lower bound theorem (Theorem 2). Hence $E[T] = E[T_Y] \geq \frac{\ln(s_0)-\ln(s_{\text{aim}})}{\delta}\frac{1-\beta}{1+\beta}$. Elementary algebra shows $(1-\beta) \geq (1-2\beta)(1+\beta)$, which gives the second, more convenient lower bound. □

Corollary 5 *Assume that the assumptions of Theorem 4 are satisfied, however with δ replaced by $\delta(s)$ for some function $\delta : S \to (0, 1]$. Then for any $s_{\text{aim}} < \tilde{s}_0 \leq s_0$, we have*

$$E[T] \geq \frac{\ln(\tilde{s}_0) - \ln(s_{\text{aim}})}{\delta_{\max}(\tilde{s}_0)}(1-2\beta),$$

where $\delta_{\max}(\tilde{s}_0) := \max\{\delta(s) \mid s_{\text{aim}} < s \leq \tilde{s}_0\}$.

Proof Let $\tilde{S} := S \cap [0, \tilde{s}_0]$. Let $\tilde{g} : \Omega \to \tilde{S}; \omega \mapsto \min\{\tilde{s}_0, g(\omega)\}$. Let $\omega \in \Omega$, $s_{\text{aim}} < s \leq \tilde{g}(\omega)$, and t be such that $\Pr[X^{(t)} = \omega] > 0$. Then

$$\begin{aligned} E[(s - \tilde{g}(X^{(t+1)}))_+ \mid X^{(t)} = \omega] &= E[(s - g(X^{(t+1)}))_+ \mid X^{(t)} = \omega] \\ &\leq \delta(s)s \leq \delta_{\max}(\tilde{s}_0)s \end{aligned}$$

by the assumptions of Theorem 4 and $s \leq \tilde{s}_0$. Similarly,

$$\begin{aligned} \Pr[s - \tilde{g}(X^{(t+1)}) \geq \beta s \mid X^{(t)} = \omega] &= \Pr[s - g(X^{(t+1)}) \geq \beta s \mid X^{(t)} = \omega] \\ &\leq \frac{\beta\delta(s)}{\ln(s)} \leq \frac{\beta\delta_{\max}(\tilde{s}_0)}{\ln(s)}. \end{aligned}$$

Hence we may apply Theorem 4 to $(\tilde{S}, \tilde{s}_0, \tilde{g}, \delta_{\max}(\tilde{s}_0))$ instead of (S, s_0, g, δ) and obtain the claimed bound. □

4 Mutation Strength Chosen Uniformly at Random

In this section we analyze the mutation operator with uniform mutation strength, that is, if the mutation operator chooses to change a position, it resets the current value to a different value chosen independently (for each position) and uniformly at random. We shall prove the same results, tight apart from lower order terms, for all r-valued OneMax functions defined in Sect. 2. Let f be one such objective function and let z be its target.

When regarding a single component x_i of the solution vector, it seems that replacing a non-optimal x_i by some y_i that is closer to the target, but still different from it, gains us some fitness, but does not lead to a structural advantage (because we still need an elementary mutation that resets this value exactly to the target value z_i). This intuitive feeling is correct for RLS and not correct for the $(1+1)$ EA.

4.1 RLS with Uniform Mutation Strength

For RLS, we turn the above intuition into the potential function $g : [0 \ldots r-1]^n \to \mathbb{R}; x \mapsto H(x, z) = |\{i \in [n] \mid x_i \neq z_i\}|$, the Hamming distance, which counts the number of non-optimal positions in the current solution x. We get both an upper and a lower bound on the drift in this potential which allow us to apply multiplicative drift theorems. From that we get the following result.

Theorem 6 *Let f be any r-valued* OneMax *function with target $z \in [0 \ldots r-1]^n$. Then randomized local search (RLS) with uniform mutation strength has an optimization time T satisfying*

$$E[T] = n(r-1)(\ln(n) + \Theta(1)).$$

If x_0 denotes the random initial individual, then for all $x \in [0 \ldots r-1]^n$ we have

$$E[T \mid x_0 = x] = n(r-1) H_{H(x,z)},$$

where, for any positive integer k, we let $H_k := \sum_{j=1}^{k} 1/j$ denote the k-th Harmonic number.

Proof Consider one iteration of RLS started with a current solution $x \neq z$. Let y be the current solution after one iteration, that is, the value of x after mutation and selection. We observe that $g(y) = g(x) - 1$ if and only if the mutation operator selects a non-optimal position i of x (this happens with probability $g(x)/n$) and then replaces x_i by z_i (this happens with probability $1/(r-1)$). In all other cases, we have $g(y) = g(x)$, though not necessarily $y = x$. Consequently, the expected progress with respect to g in this iteration is

$$g(x) - E[g(y)] = \frac{g(x)}{n(r-1)}. \tag{1}$$

Let us denote by T_{x_0} the run time of RLS conditional on the initial search point being x_0. Then the multiplicative drift theorem (Theorem 1) gives an upper bound of

$$E[T_{x_0}] \leq n(r-1)(\ln(g(x_0)) + 1).$$

Similarly, by (1) and the fact that RLS touches only one position of the solution vector, the assumptions of the multiplicative drift theorem for lower bounds (Theorem 2) are easily seen to be satisfied with $\delta = \frac{1}{n(r-1)}$, $s_{\text{aim}} = \ln n$ and $\beta = 1/\ln n$. Consequently, assuming $g(x_0) = \exp(\omega(\ln \ln n))$ in the second estimate, we obtain

$$\begin{aligned} E[T_{x_0}] &\geq n(r-1)(\ln(g(x_0)) - \ln \ln n)(1 - 2/\ln(n)) \\ &= n(r-1)\ln(g(x_0))(1 - o(1)). \end{aligned}$$

In the above analysis we used multiplicative drift with the Hamming distance because this in a generic manner gave a very strong result. We also used a drift approach to ease the comparison with the other results we will obtain, also via drift analysis. For this particular problem, also a very problem-specific approach can be used, which gives an even sharper result. Consider a run of RLS starting with a search point x_0. For $i \in [0 \ldots g(x_0)]$, let T_i denote the first iteration after which $g(x) \leq i$, where

$T_{g(x_0)} = 0$. Then equation (1) shows that $E[T_{i-1} - T_i] = \frac{n(r-1)}{i}$ for all $i \in [g(x_0)]$. Consequently,

$$E[T_{x_0}] = E\left[\sum_{i=1}^{g(x_0)} (T_{i-1} - T_i)\right] = \sum_{i=1}^{g(x_0)} E[T_{i-1} - T_i]$$
$$= n(r-1) \sum_{i=1}^{g(x_0)} \frac{1}{i} = n(r-1) H_{g(x_0)},$$

where for all $k \in \mathbb{N}$, $H_k := \sum_{i=1}^{k} (1/i)$ is the kth Harmonic number. The harmonic number is well-understood, e.g., we have $H_k = \ln(k) + \gamma + O(1/k)$ with $\gamma = 0.5772\ldots.$ being the Euler–Mascheroni constant and we have the non-asymptotic bounds $\ln(k) \leq H_k \leq \ln(k) + 1$, which gives

$$n(r-1)\ln(g(x_0)) \leq E[T_{x_0}] \leq n(r-1)(\ln(g(x_0)) + 1).$$

By the law of total probability, the expected run time of RLS (with the usual random initialization) is $E[T] = n(r-1)E[H_{g(x_0)}]$. The expected potential of the random initial search point is $E[g(x_0)] = n(1 - 1/r)$. By a Chernoff bound (e.g., Theorem 1.11 in [8]), we see that $\Pr[|g(x_0) - E[g(x_0)]| \geq \sqrt{n \ln n}] \leq 2n^{-2}$. Hence $E[H_{g(x_0)}] \leq (1 - 2n^{-2}) H_{\lfloor E[g(x_0)] + \sqrt{n \ln n} \rfloor} + 2n^{-2} H_n \leq H_{\lfloor E[g(x_0)] \rfloor} + \lceil \sqrt{n \ln n} \rceil \frac{1}{E[g(x_0)]} + O(n^{-2} \ln n) = \ln(n) + O(1) + O(\sqrt{\ln(n)/n}) + O(n^{-2} \ln n)$. Similarly, we see $E[H_{g(x_0)}] \geq \ln(n) - O(1)$. The error terms could be further reduced by arguments as used in [9], where for the case $r = 2$ a run time bound of $E[T] = nH_{n/2} - 1/2 \pm o(1)$ was shown. We do not detail this idea any further. □

4.2 The (1+1) EA with Uniform Mutation Strength

We now consider the same run time analysis problem for the (1+1) EA, that is, instead of selecting a single random entry of the solution vector and applying an elementary mutation to it, we select each entry independently with probability $1/n$ and mutate all selected entries. Our main result is the following.

Theorem 7 *For any r-valued* ONEMAX *function, the* (1+1) *EA with uniform mutation strength has an expected optimization time of*

$$E[T] = e(r-1)n\ln(n) + o(nr\log n).$$

As we will see, since several entries can be changed in one mutation step, the optimization process now significantly differs from the RLS process. This has two important consequences. First, while for the RLS process the Hamming distance of the current search point precisely determined the expected remaining optimization time, this is not true anymore for the (1 + 1) EA. This can be seen (with some mild calculations which we omit here) from the search points $x = (r, 0, \ldots, 0)$ and $y = (1, 0, \ldots, 0)$ and the fitness function f defined by $f(x) = \sum_{i=1}^{n} x_0$.

The second, worse, consequence is that the Hamming distance does not lead to a positive drift from each search point. Consider again $x = (r, 0, \dots, 0)$ and f as above. Denote by x' the search point after one mutation-selection cycle started with x. Let g be the Hamming distance to the optimum $x^* = (0, \dots, 0)$ of f. Then, for $r \geq 5$, the drift from the search point x satisfies $E[g(x) - g(x')] \leq -(1 \pm o(1))\frac{r-4}{2e(r-1)n} < 0$. Indeed, we have $g(x') = 0$, that is, $g(x) - g(x') = 1$, with probability $(1-1/n)^{n-1}(1/n)(1/(r-1)) = (1 \pm o(1))\frac{1}{e(r-1)n}$. This is the only event that gives a positive drift. On the other hand, with probability at least $(1-1/n)^{n-2}(n-1)(1/n^2)(1+2+\dots+(r-2))/(r-1)^2 = (1 \pm o(1))\frac{r-2}{2e(r-1)n}$, the mutation operator touches exactly the first and one other entry of x and does so in a way that the first entry does not become zero and the second entry remains small enough for x' to be accepted. This event leads to a drift of -1, showing the claim.

For these reasons, we resort to the actual fitness as potential function in our upper bound proof (proof of Theorem 8). It is clear that the fitness also is not a perfect measure for the remaining optimization time (compare, e.g., the search points $(2, 0, \dots, 0)$ and $(1, 1, 0, \dots, 0)$), but naturally we have a positive drift from each non-optimal search point, which we shall exploit via the variable drift theorem. For the lower bound, a worsening of the Hamming distance in the optimization process is less of a problem, since we only need an upper bound for the drift. Hence for the lower bound, we can use multiplicative drift with g again. However, since the process may move backwards occasionally, we cannot apply Witt's lower bound drift theorem (Theorem 2), but have to prove a variant of it that does not require that the process only moves forward. This lower bound theorem for multiplicative drift might be of interest beyond this work.

4.2.1 An Upper Bound for the Run Time

Theorem 8 *For any r-valued* OneMax *function f, the $(1+1)$ EA with uniform mutation strength has an expected optimization time of*

$$\begin{aligned} E[T] &\leq e(r-1)n\ln(n) + (2+\ln(2))e(r-1)n \\ &= e(r-1)n\ln(n) + O(rn). \end{aligned}$$

Proof Let z be the optimum of f. Then f can be written as $f(x) = \sum_{i=1}^{n} d(x_i, z_i)$, where d is one of the distance measures on $[0 \dots r-1]$ that were described in Sect. 2. Let x be a fixed search point and y be the result of applying one mutation and selection step to x. We use the short-hand $d_i := d(x_i, z_i)$. We first show that

$$\Delta := f(x) - E[f(y)] \geq \frac{1}{2e(r-1)n}\sum_{i=1}^{n} d_i(d_i+1). \tag{2}$$

Indeed, $f(x) - f(y)$ is always non-negative. Consequently, it suffices to point out events that lead to the claimed drift. With probability $(1-(1/n))^{n-1} \geq (1/e)$, the mutation operator changes exactly one position of x. This position then is uniformly distributed in $[n]$. Conditional on this position being i, we have $\Delta \geq \sum_{\delta=1}^{d_i} \delta/(r-1) =$

$\frac{d_i(d_i+1)}{2(r-1)}$, where the first inequality uses the fact that all our fitness functions are of the type that if there is a value $x_i \in [0 \dots r-1]$ with $d(x_i, z_i) = k$, then for each $j \in [0 \dots k-1]$ there is at least one value $y_i \in [0 \dots r-1]$ such that $d(y_i, z_i) = j$. This shows (2).

For any $d \geq 1$, we have $d(d+1) \geq 2d$ and $d(d+1) \geq d^2$. Also, the mapping $d \mapsto d^2$ is convex. Consequently, we have $\Delta \geq \frac{1}{2e(r-1)n^2} f(x)^2$ and $\Delta \geq \frac{1}{e(r-1)n} f(x)$, that is, $\Delta \geq \max\{\frac{1}{2e(r-1)n^2} f(x)^2, \frac{1}{e(r-1)n} f(x)\}$. To this drift expression, we apply Johannsen's [33] variable drift theorem (Theorem 3). Let $S = [0 \dots (r-1)n]$. Let $h : \mathbb{R}_{>0} \to \mathbb{R}_{>0}$ be defined by $h(s) = \frac{1}{2e(r-1)n^2} s^2$ for $s \geq 2n$ and $h(s) = \frac{1}{e(r-1)n} s$ for $s < 2n$. Then h is a continuous increasing function satisfying $\Delta \geq h(f(x))$. Consider the process $X_0, X_1, \dots$ with X_t describing the fitness after the tth iteration. Given that we start with a fitness of X_0, Johannsen's drift theorem gives

$$\begin{aligned}
E[T] &\leq \frac{1}{h(1)} + \int_1^{X_0} \frac{1}{h(s)} ds \\
&= e(r-1)n + \int_{2n}^{X_0} 2e(r-1)\frac{n^2}{s^2} ds + \int_1^{2n} e(r-1)\frac{n}{s} ds \\
&\leq e(r-1)n + 2e(r-1)n^2 \left(\frac{1}{2n} - \frac{1}{X_0}\right) + e(r-1)n \ln(2n) \\
&\leq e(r-1)n \ln(n) + (1 + 1 + \ln(2)) e(r-1)n.
\end{aligned}$$

□

We may remark that the drift estimate above is pessimistic in that it applies to all r-valued OneMax functions. For an r-valued OneMax function using the ring metric or one having the optimum close to $(r/2, \dots, r/2)$, we typically have two different bit values in each positive distance from z_i. In this case, the drift stemming from exactly position i being selected for mutation is $\Delta \geq d_i/(r-1) + \sum_{\delta=1}^{d_i-1} 2\delta/(r-1) = \frac{d_i^2}{r-1}$, that is, nearly twice the value we computed above. The fact that in the following subsection we prove a lower bound matching the above upper bound for all r-valued OneMax functions shows that this, almost twice as high, drift has an insignificant influence on the run time.

4.2.2 A Lower Bound for the Run Time

We aim at proving a lower bound, again via drift analysis, that is, via transforming an upper bound on the expected progress (with respect to a suitable potential function) into a lower bound on the expected run time. Since we only need an upper bound on the progress, we can again (as in the RLS analysis) work with the Hamming distance $g(x) = H(x, z)$ to the optimum z as potential and, in the upper estimate of the drift, ignore the fact that this potential may increase. The advantage of working with the Hamming distance is that the drift computation is easy and we observe multiplicative drift, which is usually convenient to work with.

We want to apply the multiplicative drift theorem given in Theorem 4. To this end, we will compute that the Hamming distance to the optimum satisfies the assumptions of our drift results in the following lemma, and afterwards state and prove the precise lower bound.

Lemma 9 *Let f be an r-valued* OneMax *function with optimum z. Let $x \in [0 \dots r-1]^n$ and y be the outcome of applying mutation and selection to x. Let $s^+ := H(x,z)$ and $s \le s^+$. Then*

$$E[(s - H(y,z))_+] \le \frac{s}{e(r-1)n}\left(\frac{1}{1-1/n} + 3e\frac{s+1}{(r-1)n}\right).$$

Proof Let u be the outcome of mutating x with uniform mutation strength and y be the result of applying selection (with respect to f) to x and u.

We consider first the case that $s = s^+$. With probability $(1-(1/n))^{n-1}$, u and x differ in exactly one position. Conditional on this, $E[(s - H(y,z))_+] = s/(r-1)n$. The only other event in which possibly $s > H(y,z)$ is that u and x differ in at least two positions i and j such that $u_i = z_i$ and $u_j = z_j$. The probability for this to happen is $1 - (1 - 1/(r-1)n)^s - (s/(r-1)n)(1 - 1/(r-1)n)^{s-1} \le 1 - (1 - s/(r-1)n) - (s/(r-1)n)(1 - (s-1)/(r-1)n) = s(s-1)/(r-1)^2n^2$. In this case, we can estimate $(s - H(y,z))_+$ from above by the number of non-correct positions that are touched by the mutation operator, which in this case is $\text{Bin}(s, 1/n)$ conditional on being at least two, which again is at most 3. Consequently, in the case that $s = s^+$, we have $E[(s - H(y,z)_+] \le (1-(1/n))^{n-1}s/(r-1)n + 3s(s-1)/(r-1)^2n^2 \le (s/(r-1)n)(1/e(1-1/n) + 3(s-1)/(r-1)n)$.

Let now $s \le s^+ - 1$. Let $Z := |\{i \in [n] \mid x_i \ne z_i \ne u_i\}| \in [0 \dots s^+]$ be the number of positions that are incorrect in both x and u. Clearly, $H(y,z)$ stochastically dominates Z, which we write as $H(x,z) \succeq Z$. Let Z' be defined analogous to Z but for an original search point x' with $H(x',z) = s+1 \le H(x,z)$. Then, clearly, $Z \succeq Z'$. Consequently, $s - H(y,z) \preceq s - Z \preceq s - Z'$, and consequently, $(s - H(y,z))_+ \preceq (s - Z')_+$ and $E[(s - H(y,z)_+] \le E[(s - Z')_+]$. The only way to get a positive value for $s - Z'$ is that at least two incorrect positions of x' are changed to their correct value in the mutation offspring. Analogous to the previous paragraph, the probability for this to happen is $1 - (1 - 1/(r-1)n)^{s+1} - ((s+1)/(r-1)n)(1 - 1/(r-1)n)^s \le (s+1)s/(r-1)^2n^2$. In this case, we can estimate $(s - Z')_+$ from above by the number of incorrect positions that are touched by the mutation operator (conditional on being at least two) minus one, which is at most 2. We conclude $E[(s - H(y,z))_+] \le E[(s - Z')_+] \le 2(s+1)s/(r-1)^2n^2$.

Putting the two cases together, we see that we always have $E[(s - H(y,z))_+] \le (s/(r-1)n)(1/e(1-1/n) + 3(s+1)/(r-1)n)$. □

Lemma 10 *Let f be an r-valued* OneMax *function with optimum z. Let $s_{\text{aim}} = \ln(n)^3$ and $\beta = 1/\ln(n)$. Let $x \in [0 \dots r-1]^n$ with $H(x,z) > s_{\text{aim}}$. Let $s_{\text{aim}} < s \le H(x,z)$. Let y be the outcome of applying mutation and selection to x. Then* $\Pr[s - H(y,z) \ge \beta s] \le \frac{1}{r-1}2^{-\ln(n)^2}$ *if $n \ge 11$.*

Proof We have that $\Pr[s - H(y,z) \ge \beta s] \le \Pr[H(x,z) - H(y,z) \ge \beta s_{\text{aim}}]$. The latter is at most the probability that at least $\beta s_{\text{aim}} = \ln(n)^2$ positions of x flip to a particular value (namely the one given by z) in one mutation step. Since the expected number of positions flipping to the correct value is at most $1/(r-1)$, a strong multiplicative Chernoff bound (e.g., Cor. 1.10(b) in [8]) shows that this number is greater than $\ln(n)^2$ with probability at most $(e/\ln(n)^2(r-1))^{\ln(n)^2} \le \frac{1}{r-1}2^{-\ln(n)^2}$ for $n \ge 10.29 \approx \exp(\sqrt{2e})$. □

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

Theorem 11 *For any r-valued* OneMax *function, the* $(1+1)$ *EA with uniform mutation strength has an expected optimization time of*

$$\begin{aligned} E[T] &\ge e(r-1)n\,(\ln(n) - 6\ln\ln(n))\,(1 - O(1/\ln(n))) \\ &\ge e(r-1)n\ln(n) - O((r-1)n\ln\ln(n)). \end{aligned}$$

Proof Let n be sufficiently large. Let $\Omega = [0\ldots r-1]^n$ and $f : \Omega \to \mathbb{R}$ an r-valued OneMax function with optimum z. Let $s_{\text{aim}} = \ln(n)^3$ and $\beta = 1/\ln(n)$. For all $s_{\text{aim}} < s \le n$, let $\delta(s) := (1/e(r-1)n)(1/(1-1/n) + 3e(s+1)/(r-1)n)$. Consider a run of the $(1+1)$ EA optimizing f initialized with a random search point $X^{(0)}$. We have $E[H(X^{(0)}, z)] = n(1-1/r)$. Consequently, we have $H(X^{(0)}, z) \ge n/3$ with probability $1 - \exp(-\Omega(n))$. In the following, we thus assume that $X^{(0)}$ is a fixed initial search point such that $H(X^{(0)}, z) \ge n/3$. Denote by $X^{(t)}$ the search point building the one-element population of this EA after the t-th iteration. Let $g : \Omega \to \mathbb{N}; x \mapsto H(x, z)$. By Lemma 9 and 10, the following conditions are satisfied for all $\omega \in \Omega$, all $s_{\text{aim}} < s \le g(\omega)$, and all $t \ge 0$ with $\Pr[X^{(t)} = \omega] > 0$.

$$E\left[(s - g(X^{(t+1)}))_+ \mid X^{(t)} = \omega\right] \le \delta(s)s.$$

$$\Pr\left[s - g(X^{(t+1)}) \ge \beta s \mid X^{(t)} = \omega\right] \le \frac{\beta\delta(s)}{\ln(s)}.$$

We apply Corollary 5 with $\tilde{s}_0 = n/\ln(n)^3 \le n/3$ and $\delta_{\max}(\tilde{s}_0) \le \frac{1}{e(r-1)n}(1 + O(1/n) + O(1/\ln(n)^3(r-1)))$ and obtain

$$\begin{aligned} E[T] &\ge \frac{\ln(\tilde{s}_0) - \ln(s_{\text{aim}})}{\delta_{\max}(\tilde{s}_0)}(1 - 2\beta) \\ &\ge e(r-1)n\ln(n) - O((r-1)n\ln\ln(n)). \end{aligned}$$

□

We remark that the lower order term $O((r-1)n\log\log n)$ in this lower bound could be removed with stronger methods. We preferred to use the simple and natural proof approach via multiplicative drift, because it is easy to handle and still relatively precisely describes the true behavior of the process. As is visible from Lemma 9, in the early stages the progress is slightly faster than the multiplicative (main) term $s/e(r-1)n$. This is why we cut out the regime from the initial H-value of approximately $n(1-1/r)$ up to an H-value of $\tilde{s}_0 = n/\ln(n)^3$, resulting in a $-\Theta((r-1)n\log\log n)$ term in our lower bound. Another $-\Theta((r-1)n\log\log n)$ term stems from the second condition of Witt's lower bound drift theorem (which is similar to the second condition of our theorem). To prove a bound sharp up to terms of order $(r-1)n \log\log n$, we need $\beta \leq \log\log n/\log n$. However, this forbids using an s_{aim} smaller than $1/\beta = \log n/\log\log n$, since otherwise any improvement would count into the bad event of the second condition. An s_{aim} of at least polylogarithmic size immediately implies an $\Omega((r-1)n \log\log n)$ additive distance to the upper bound proven in Theorem 8. We are very optimistic that via variable drift, in particular, the lower bound theorem of [16], both difficulties could be overcome. We do not think that this small improvement justifies the effort, though.

5 Unit Mutation Strength

In this section we regard the mutation operator that applies only ± 1 changes to each component.

It is not very surprising that RLS with the ± 1 variation operator needs $\Theta(n(r+\log n))$ fitness evaluations in expectation to optimize any r-valued OneMax function. In fact, the proof is similar to the analysis of the (1 + 1) EA equipped with the ± 1 variation operator (the proof for the (1 + 1) EA is a bit more involved). We will make use of the following observation. There are two extreme kinds of individuals with fitness n. The first kind is only incorrect in one position (by an amount of n); the second kind is incorrect in every position (by an amount of 1). The first kind of individual is hard to improve (the deficient position has to be chosen for variation), while the second kind is very easy to improve (every position allows for improvement). We reflect this in our choice of potential function by giving each position a weight exponential in the amount that it is incorrect, and then sum over all weights.

Theorem 12 *The expected optimization time of RLS with the* ± 1 *variation operator is* $\Theta(n(r+\log n))$ *for any* r*-valued* OneMax *function.*

Proof The lower bound $\Omega(nr)$ is quite immediate: with probability 1/2 we start in a search point of fitness at most $nr/2$ and in each step the algorithm increases the fitness by at most one. On the other hand, there is a coupon collector effect which yields the $\Omega(n \log n)$ lower bound. Indeed, it is well-known that this is the expected number of RLS iterations that we need in case of $r=2$, and larger values of r will only delay optimization.

We now turn to the more interesting upper bound. Let any r-valued OneMax function be given with metric d and target z. We want to employ a multiplicative drift theorem (see Theorem 1). We measure the potential of a search point by the following drift function. For all $x \in \Omega = [0 \dots r-1]^n$, let

$$g(x) := \sum_{i=1}^{n} \left(w^{d(z_i, x_i)} - 1 \right), \tag{3}$$

where $w := 1+\varepsilon$ is an arbitrary constant between 1 and 2. In fact, for the analysis of RLS we can simply set $w := 2$ but since we want to re-use this part in the analysis of the (1 + 1) EA, we prefer the more general definition here.

We regard how the potential changes on average in one iteration. Let x denote the current search point and let y denote the search point that we obtain from x after one iteration of RLS (after selection). Clearly, we have that each position is equally likely to be selected for variation. When a non-optimal component i is selected, then the probability that y_i is closer to z_i than x_i is at least $1/2$, while for every already optimized component we will not accept any move of RLS (thus implying $y_i = x_i$). This shows that, abbreviating $d_i := d(z_i, x_i)$ for all $i \in [n]$, and denoting by $O := \{i \in [n] \mid x_i = z_i\}$ the set of already optimized bits,

$$\begin{aligned} E[g(x) - g(y) \mid x] &= \tfrac{1}{2n} \sum_{i \in [n] \setminus O} \Big((w^{d_i} - 1) - (w^{d_i - 1} - 1) \Big) \\ &= \tfrac{1}{2n} \sum_{i \in [n] \setminus O} \left(1 - \tfrac{1}{w}\right) w^{d_i} \\ &\geq \tfrac{1}{2n} \left(1 - \tfrac{1}{w}\right) \sum_{i \in [n]} \Big(w^{d_i} - 1 \Big) \\ &= \tfrac{1}{2n} \left(1 - \tfrac{1}{w}\right) g(x). \end{aligned}$$

Furthermore, the maximal potential that a search point can obtain is at most nw^r. Plugging all this into the multiplicative drift (see Theorem 1), we see that the expected optimization time is of order at most $\ln(nw^r)/\left(\frac{1}{2n}(1 - \frac{1}{w})\right) = O(n(r + \log n))$, as desired. □

For the analysis of the (1 + 1) EA we will proceed similarly as for RLS. To help with the added complexity, we use the following lemma.

Lemma 13 *Let n be fixed, let q be a cost function on* elements *of* $[n]$ *and let c be a cost function on* subsets *of* $[n]$. *Furthermore, let a random variable S ranging over subsets of* $[n]$ *be given. Then we have*

$$\forall T \subseteq [n] : c(T) \leq \sum_{i \in T} q(i) \Rightarrow E[c(S)] \leq \sum_{i=1}^{n} q(i) \Pr[i \in S]; \tag{4}$$

and

$$\forall T \subseteq [n] : c(T) \geq \sum_{i \in T} q(i) \Rightarrow E[c(S)] \geq \sum_{i=1}^{n} q(i) \Pr[i \in S]. \tag{5}$$

Proof We have $E[c(S)] = \sum_{T \subseteq [n]} \Pr[S = T] c(S) \leq \sum_{T \subseteq [n]} \Pr[S = T] \sum_{i \in T}^{n} q(i) = \sum_{i=1}^{n} q(i) \Pr[i \in S]$. The other direction follows analogously. □

The proof for the case of the (1 + 1) EA follows along similar lines, but is (significantly) more involved.

Theorem 14 *The expected optimization time of the* (1 + 1) *EA with the* ±1 *variation operator is* $\Theta(n(r + \log n))$ *for any r-valued* OneMax *function.*

Proof The lower bound $\Omega(nr)$ follows almost as for RLS: With constant probability the initial search point is $\Theta(nr)$ away from the optimum, and the expected progress towards the optimum is bounded form above by 1. Thus, with a simple lower-bound additive drift theorem [28], the lower bound of $\Omega(nr)$ follows.

Regarding the upper bound, let any r-valued OneMax function be given with metric d and target z. We want to employ multiplicative drift again. We fix some $w > 1$ to be specified later. With any search point $x \in \Omega$ we associate a vector $d \in \mathbb{R}^n$ such that, for all $i \le n$, $d_i = d(x_i, z_i)$. We use the same potential g on Ω as for the analysis of RLS, that is, for all $x \in \Omega$,

$$g(x) = \sum_{i=1}^{n} (w^{d_i} - 1).$$

Let any current search point $x \in \Omega$ be given and let Y be the random variable describing the search point after one cycle of mutation and selection. Let E_1 be the event that Y is obtained from x by flipping exactly one bit and the result is accepted (that is, $f(Y) \le f(x)$). Let E_2 be the event that at least 2 bits flip and the result is accepted. The total drift in the potential g is now $E[g(x) - g(Y)] = E[g(x) - g(Y) \mid E_1]\Pr[E_1] + E[g(x) - g(Y) \mid E_2]\Pr[E_2]$. We are now going to estimate $E[g(x) - g(Y) \mid E_2]$. The random variable Y is completely determined by choosing a set $S \subseteq [n]$ of bit positions to change in x and then, for each such position $i \in S$, choosing how to change it (away or towards the target z_i). For each choice $S \subseteq [n]$, let $A(S)$ be the set of all possible values for Y which have the set S as positions of change. For each possible $S \subseteq [n]$, let $Y(S)$ be the random variable Y conditional on making changes exactly at the bit positions of S. Thus, we can now write the random variable Y as

$$Y = \sum_{S} Y(S) \Pr[S].$$

We are now going to estimate, for any possible S,

$$E[g(Y(S)) - g(x)].$$

For all possible S, let $c(S) = E[g(Y(S)) - g(x)]$. Let a possible S be given. Note that $Y(S)$ is the uniform distribution on $A(S)$. For each $y \in A(S)$ and each $i \in S$, we have $d(y_i, z_i) - d_i \in \{-1, 0, 1\}$ (note that the case of being 0 can only occur when r is odd and we have a circle, or in other such border cases); in the case that this value is 1 we call (y, i) an *up-pair*, and in case that this value is -1 we call this pair a *down-pair*. We let U be the set of all up-pairs. As we only consider accepted mutations, we have that, for all $y \in A(S)$, $\sum_{i \in S} d(y_i, z_i) - d_i \le 0$. This implies that there are at least as many down-pairs as there are up-pairs in $A(S) \times S$. Furthermore, for any up-pair (y, i) with $d_i \neq 0$ there is $y' \in A(S)$ such that (y', i) is a down-pair and, for all $j \in S \setminus \{i\}$, $y'_j = y_j$. Thus, for all up-pairs $(y, i) \in U$ there is a down-pair $(\overline{y}, \overline{i})$, such that the mapping $(y, i) \mapsto (\overline{y}, \overline{i})$ is injective and, for all $(y, i) \in U$ with $d_i \neq 0$, $(\overline{y}, \overline{i}) = (y', i)$. Note that, for all up-pairs (y, i), we have $d_i \le d_{\overline{i}}$.

We now get, for any $(y, i) \in U$,

$$w^{d(y_i,z_i)} - w^{d_i} + w^{d(\overline{y}_{\overline{i}},z_{\overline{i}})} - w^{d_{\overline{i}}} \leq w^{d_i}\left(w - 1 + \frac{1}{w} - 1\right) = w^{d_i}\frac{(w-1)^2}{w}.$$

Overall we have

$$\begin{aligned} c(S) = E[g(Y(S)) - g(x)] &= \frac{1}{|A(S)|}\sum_{y \in A(S)} g(y) - g(x) \\ &= \frac{1}{|A(S)|}\sum_{y\in A(S)}\sum_{i=1}^{n}\left(w^{d(y_i,z_i)} - w^{d_i}\right) \\ &\leq \frac{1}{|A(S)|}\sum_{(y,i)\in U}\left(w^{d(y_i,z_i)} - w^{d_i} + w^{d(\overline{y}_{\overline{i}},z_{\overline{i}})} - w^{d_{\overline{i}}}\right) \\ &\leq \frac{1}{|A(S)|}\sum_{(y,i)\in U} w^{d_i}\frac{(w-1)^2}{w} \leq \frac{1}{2}\sum_{i\in S} w^{d_i}\frac{(w-1)^2}{w}. \end{aligned}$$

Using Lemma 13, we see that

$$E[g(Y) - g(x) \mid E_2] \leq \sum_{i=1}^{n}\frac{1}{n}w^{d_i}\frac{(w-1)^2}{2w} = \frac{(w-1)^2}{2wn}\sum_{i=1}^{n} w^{d_i}.$$

We use the following estimation of progress we can make with changing exactly one position.

$$E[g(x) - g(Y) \mid E_1]\Pr[E_1] \geq \frac{1}{2ne}\sum_{i\in[n]}\left(1 - \tfrac{1}{w}\right)w^{d_i} = \frac{w-1}{2wne}\sum_{i=1}^{n} w^{d_i}.$$

Let w be any constant > 1 such that $w - 1 - e(w-1)^2 > 0$, and let $c = (w - 1 - e(w-1)^2)/e$. Then we have

$$\begin{aligned} E[g(x) - g(Y)] &\geq \frac{w-1}{2wne}\sum_{i=1}^{n} w^{d_i} - \frac{(w-1)^2}{2wn}\sum_{i=1}^{n} w^{d_i} \\ &= \frac{w - 1 - e(w-1)^2}{2wne}\sum_{i=1}^{n} w^{d_i} = \frac{c}{2wn}\sum_{i=1}^{n} w^{d_i} \geq \frac{c}{2wn}g(x). \end{aligned}$$

Again, the maximal potential that a search point can obtain is at most nw^r. Plugging all this into the multiplicative drift (see Theorem 1), we see that the expected optimization time is of order at most $\ln(nw^r)/\left(\frac{c}{2wn}\right) = O(n(r + \log n))$, as desired. $\square$

6 Harmonic Mutation Strength

In this section we will consider a mutation operator with variable step size. The idea is that different distances to the target value require different step sizes for rapid progress. We consider a mutation operator which, in each iteration, chooses its step size from a fixed distribution. As distribution we use what we call the *harmonic distribution*, which chooses step size $j \in [1 \dots r-1]$ with probability proportional to $1/j$. Using the bound on the harmonic number $H_{r-1} < 1 + \ln r$, we see that the probability of choosing such a j is at least $1/(j(1+\ln r))$.

Theorem 15 *The RLS as well as the* $(1+1)$ *EA with the harmonically distributed step size (described above) has an expected optimization time of* $\Theta(n \log r(\log n + \log r))$ *on any* r*-valued* OneMax *function.*

Proof We first show the upper bound by considering drift on the fitness. Let any $x \in \Omega$ be given, let Y be the random variable describing the best individual of the next iteration and let $A_{i,j}$ be the event that Y differs from x in exactly bit position i and this bit position is now j closer to the optimum. Note that, for both RLS and the $(1+1)$ EA, we get $\Pr[A_{i,j}] \geq \frac{1}{2enj(1+\ln r)}$. We have

$$\begin{aligned} E[f(x) - f(Y)] &\geq \sum_{i=1}^{n} \sum_{j=1}^{d_i} E[f(x) - f(Y) \mid A_{i,j}] \Pr[A_{i,j}] = \sum_{i=1}^{n} \sum_{j=1}^{d_i} j \Pr[A_{i,j}] \\ &\geq \sum_{i=1}^{n} \sum_{j=1}^{d_i} \frac{j}{2enj(1+\ln r)} = \sum_{i=1}^{n} \frac{d_i}{2en(1+\ln r)} \\ &= \frac{1}{2en(1+\ln r)} f(x). \end{aligned}$$

As the initial fitness is less than rn, the multiplicative drift theorem (see Theorem 1) gives us the desired total optimization time.

Now we turn to the lower bound. A straightforward coupon collector argument gives us the lower bound of $\Omega(n \log r \log n)$, since each position has to change from incorrect to correct at some point, and that mutation has a probability of $O(1/(n \log r))$. It remains to show a lower bound of $\Omega(n(\log r)^2)$. To this end, let f be any r-valued OneMax function and x^* its optimum. Let $g(x) = d(x_1, x_1^*)$ be the distance of the first position to the optimal value in the first position. Let $h(x) = \ln(g(x)+1)$. Let x' be the outcome of one mutation step and x'' be the outcome of selection from $\{x, x'\}$. We easily compute $E[\max\{0, h(x) - h(x')\}] \leq \frac{K}{n \ln r}$ for some absolute constant K. Consequently, $E[h(x) - h(x'')] \leq \frac{K}{n \ln r}$ as well. For the random initial search point, we have $g(x) \geq r/2$ with constant probability, that is, $h(x) = \Omega(\log r)$ with constant probability. Consequently, the additive drift theorem gives that the first time T at which $h(x) = 0$, satisfies $E[T] \geq \Omega(\log r)/\frac{K}{n \ln r} = \Omega(n \log^2 r)$. □

In the same way as we showed the additive drift statement $E[h(x) - h(x'')] = O(1/n \log r)$, we could have shown a multiplicative drift statement for g, namely

$E[g(x) - g(x'')] = O(g(x)/n \log r)$; in fact, the latter is implied by the former. Unfortunately, due to the presence of large jumps – we have $\Pr[g(x'') \leq g(x)/2] = \Theta(1/n \log r)$ –, we cannot exploit this via the lower bound multiplicative drift theorem.

6.1 Lower Bound for the Dependence on *r*

Naturally, the question arises whether the $O((\log r)^2)$ dependence on r can be improved. In particular, one wonders whether drawing the step size from the harmonic distribution is optimal, or whether another distribution gives a better optimization time. This is exactly the problem considered in [7], where the following result is presented, which could also be used to derive the run time bound of Theorem 15.

Theorem 16 ([7]) *Let a random process on $A = \{0, \ldots, r\}$ be given, representing the movement of a token. Fix a probability distribution of step sizes D over $\{1, \ldots, r\}$. Initially, the token is placed on a random position in A. In round t, a random step size d is chosen according to D. If the token is in position $x \geq d$, then it is moved to position $x - d$, otherwise it stays put. Let T_D be the number of rounds until the token reaches position 0. Then $\min_D(E[T_D]) = \Theta((\log r)^2)$.*

In general, our processes have a slightly different behavior (including the possibility to overshoot the goal). But for the special case of $n = 1$, the interval metric and the optimum at 0, the setting matches exactly (apart from choosing the correct direction for the step). Thus, in terms of asymptotic behavior in r, the above theorem shows that the Harmonic distribution is an optimal choice and cannot be improved.

7 Self-Adjusting Mutation Rates

In this section we consider self-adjusting mutation rates and show that they can achieve an asymptotically better performance than the static operators considered in the previous sections. For this we consider a natural generalization of RLS to a multi-valued algorithm $\texttt{RLS}_{a,b}$ with a self-adjusting mutation strength whose update rules are parametrized by the constants $1 < a \leq 2$ and $1/2 < b < 1$. The algorithm is summarized in Algorithm 3 and will be described in detail in Sect. 7.2. Before we go into the technical details, we briefly discuss in Sect. 7.1 some related work on self-adjusting parameter choices.

7.1 Adaptive Parameter Choices

In continuous optimization one easily observes that static parameter choices are not very meaningful. This is why for such problems several examples exist where adaptive parameter choices are well-understood also from a theoretical perspective (for example, the works [2,27,29] analyze the convergence rates of different evolution strategies). As has been noted in Sect. 1.4, however, such results are difficult to compare to performance guarantees in discrete optimization let alone being transferable to such problems. We recall that this is mostly due to the fact that in discrete optimization we do not study the speed of convergence but the time needed to hit an optimal solution. But even if one studies continuous optimization with an a-priori fixed target precision (see [30] and the references therein), then typically the norms used to evaluate a solution differ from the typically regarded 1-norm used in discrete optimization.

For the discrete domain, several empirical works exist that suggest an advantage of adaptive parameter updates (cf. [24], [25, Chapter 8], and [34] for surveys). However, the first work formally showing an asymptotic gain over static parameter selection is the self-adjusting choice of the population size of the $(1 + (\lambda, \lambda))$ GA proposed and analyzed in [10]. In that work the advantage is shown for the classic ONEMAX functions $f_z : \{0, 1\}^n \to \mathbb{R}, x \mapsto |\{1 \leq i \leq n \mid x_i = z_i\}|$.

When we include in our consideration other adaptive parameter choices,[1] a few situations exist for which an advantage over static parameter choices could be proven. All these works study the optimization of pseudo-Boolean functions $f : \{0, 1\}^n \to \mathbb{R}$. To be more precise, the only theoretical investigations of adaptive parameter choices in evolutionary algorithms for discrete optimization problems that we are aware of analyze advantages of

- a fitness-dependent mutation rate for the $(1+1)$ EA optimizing LEADINGONES [4],
- a fitness-dependent [15] and a self-adjusting [14] mutation rate for RLS optimizing ONEMAX,
- a fitness-dependent [3] and a self-adjusting [17] mutation rate for the $(1 + \lambda)$ EA optimizing ONEMAX,
- a self-adjusting choice of the number of parallel evaluations in a parallel EA [36],
- a fitness-dependent [11] and the above-mentioned self-adjusting [10] choice of the population size for the $(1 + (\lambda, \lambda))$ GA optimizing ONEMAX, and the same for the optimization of random satisfiability instances [5],
- a fitness-dependent mutation rate for immune algorithms with [47] and without [46] population, and
- a rank-based mutation rate for a $(\mu + 1)$ EA optimizing ONEMAX and functions with a "trap" that guides the search to local optima which are far away from the global ones [41].

We believe that self-adjusting parameter choices provide a possibility for significant improvement of many search heuristics, and theoretical analyses can offer guidance

[1] Following the terminology introduced in [24] and extended in [10, Section 3.1] we distinguish *adaptive* parameter choices into functionally-dependent and self-adjusting ones. While *functionally-dependent* parameter choices depend only on the current state of the algorithm, they may explicitly use absolute fitness values. Fitness- and rank-dependent mutation rates are a typical example for such *functionally-dependent* parameter choices. *Self-adjusting* parameter choices, in contrast, do not depend on absolute fitness information but rather on the success of previous iterations. This is the case of the parameter updates of the $\text{RLS}_{a,b}$ considered in this work. A typical representative of this class is the so-called one-fifth rule that is often used in evolution strategies for controlling the step size of the algorithm under consideration. Other dynamic update rules are either called *deterministic*—this is the case if there is no dependency between the parameters and the success or state of the optimization process other than the iteration count—or *self-adapting*. *Self-adaptive* algorithms code the parameters themselves into the genome of the individuals and hope to evolve good parameters during the optimization process.

for how to design such self-adjustment mechanisms. Our work shows that our mathematical toolbox, in particular drift analysis, is well-suited to analyze such systems.

For the sake of completeness we mention that a few other works in the theory of evolutionary computation literature exist that analyze dynamic (but, in contrast to the results mentioned above, non-adaptive) parameter choices. The following list summarizes these works.

– Jansen and Wegener [32] regard a $(1 + 1)$ EA with a *deterministic* choice of the mutation rate. They show that, depending on the problem, it can be advantageous or disadvantageous to use a mutation rate that depends on the iteration counter.
– Dang and Lehre regard in [6] a *self-adaptive* choice of mutation rates in a non-elitist EA. That is, in their work the mutation probability is encoded in each individual so that it is subject to a *global mutation* itself. Dang and Lehre analyze how the global mutation rate influences the expected performance of a (λ, λ)-type EA on LeadingOnes.

7.2 The Self-Adjusting RLS Variant, Main Result, and Proof Overview

In this section we consider the self-adjusting variant of RLS called $\texttt{RLS}_{a,b}$, see Algorithm 3.

Algorithm 3: $\texttt{RLS}_{a,b}$ with self-adjusting step sizes minimizing a function $f : [0..r-1]^n \to \mathbb{R}$

```
1 Initialization: Choose $v \in [1, \lfloor r/4 \rfloor]^n$ uniformly at random;
2 Choose $x \in [0..r-1]^n$ uniformly at random;
3 Optimization: for $t = 1, 2, 3, \dots$ do
4     $y \leftarrow x$;
5     Choose $i \in [n]$ uniformly at random;
6     With probability $1/2$ let $y_i \leftarrow x_i - \lfloor v_i \rfloor$ and let $y_i \leftarrow x_i + \lfloor v_i \rfloor$ otherwise;
7     if $f(y) < f(x)$ then $v_i \leftarrow \min\{av_i, \lfloor r/4 \rfloor\}$ else $v_i \leftarrow \max\{1, bv_i\}$ if $f(y) \le f(x)$ then $x \leftarrow y$
```

$\texttt{RLS}_{a,b}$ maintains a search point $x \in [0..r-1]^n$ as well as a real-valued *velocity vector* $v \in [1, \lfloor r/4 \rfloor]^n$; we use real values for the velocity to circumvent rounding problems. Both these strings are initialized uniformly at random, but it is not difficult to verify that all results shown in this paper apply to an arbitrary initialization of x and v. In one iteration of the algorithm a position $i \in [n]$ is chosen uniformly at random. The entry x_i is replaced by $x_i - \lfloor v_i \rfloor$ with probability $1/2$ and by $x_i + \lfloor v_i \rfloor$ otherwise (see below for how to deal with overstepping the endpoints of the interval $[0, r-1]$). The entries in positions $j \neq i$ are not subject to mutation. The resulting string y replaces x if its fitness is at least as good as the one of x, i.e., if $f(y) \le f(x)$ holds (recall that we regard the minimization of f). If the offspring y is strictly better than its parent x, i.e., if $f(y) < f(x)$, we increase the velocity v_i in the i-th component by multiplying it with the constant a and we decrease v_i to bv_i otherwise. The algorithm proceeds this way until we decide to stop it.

We will now discuss some technical details.

It may happen that $x_i - \lfloor v_i \rfloor < 0$ or $x_i + \lfloor v_i \rfloor > r - 1$. If we are working with the interval-metric then we assume that the algorithm does not change its current position, that is, the offspring is discarded and the velocity is not adjusted (decreasing the velocity in this case would lead to the same results). In the ring-metric we identify all values *modulo* r, i.e., we identify values $p < 0$ with $p + r$ and values $p > r - 1$ with $p - r$. Note that in the ring-metric it can happen that we decrease the fitness regardless of whether we add or subtract from x_i the value $\lfloor v_i \rfloor$. This in particular applies when the distance to the optimum is close to $r/2$, i.e. the search point is on the opposite side of the ring than the target.

Furthermore, we emphasize that the velocity vector is an element in the real interval $[1, \lfloor r/4 \rfloor]$, that is, it does not necessarily take integer values. This technicality avoids that rounding inaccuracies accumulate over several velocity adaptations. The velocity is capped at 1 (to avoid situations in which we do not move at all) and at $\lfloor r/4 \rfloor$ (to avoid too large jumps).

To further lighten the notation, we say that the algorithm *"moves in the right direction"* or *"towards the target value"* if the distance to the target is actually decreased by $\lfloor v_i \rfloor$. Analogously, we speak otherwise of a step *"away from the target"* or *"in the wrong direction"*.

Our main result is the following statement.

Theorem 17 *For constants a, b satisfying $1 < a \leq 2$, $1/2 < b \leq 0.9$, $2ab - b - a > 0$, $a + b > 2$, and $a^2 b > 1$ (one can choose, for example, $a = 1.7$ and $b = 0.9$) the expected run time of* $\texttt{RLS}_{a,b}$ *(Algorithm 3) on any generalized r-valued* OneMax *function is $\Theta(n(\log n + \log r))$. This is asymptotically best possible among all comparison-based variants of RLS and the $(1+1)$ EA.*

We argue first the lower bound of Theorem 17.

The bound of $\Omega(n \log n)$ easily follows from a coupon collector argument: note that in the initial solution there are, with high probability, $\Theta(n)$ positions i in which the value x_i does not agree with that of the target string. The algorithm has to touch each of these positions at least once, which by the well-known coupon collector theorem (cf. [1, Section 1] for an introduction to this problem) requires $\Theta(n \log n)$ iterations on average and with high probability.

The $\Omega(n \log r)$ part of the lower bound follows from the following information theoretic argument. We first observe that $\texttt{RLS}_{a,b}$ is a comparison-based algorithm. For an arbitrary comparison-based algorithm A we argue as follows. There are r^n possible target strings in total. Since A exploits only the information whether or not the offspring has a fitness value that is at least as good as that of its parent (in the decision of whether or not to replace the parent) and whether or not its fitness is strictly better (in the decision how to update the velocity), it is a *comparison-based algorithm* that uses only $\log_2(3)$ bits of information per iteration. As such it therefore needs $\Omega(\log(r^n)) = \Omega(n \log r)$ iterations in expectation to optimize any unknown r-valued OneMax function. See [23] for how to turn the latter information-theoretic consideration into a formal proof. That already for $r = 2$ every unary unbiased black-box algorithm needs $\Omega(n \log n)$ function evaluations in expectation to optimize OneMax has been shown in [37]. The class of unary unbiased black-box algorithms includes, roughly speaking, all algorithms that do not give preference to any of the n positions of the string nor to any of the possible values $[0 \ldots r-1]$ and that employ only mutation and random sampling as variation operations (that is, in particular no crossover operations are allowed for unary unbiased algorithms). It is easily verified that $\mathtt{RLS}_{a,b}$ is unary unbiased.

To prove the upper bound we use *drift analysis*; multiplicative drift analysis to be more precise. To this end, we need to find a mapping of the state (x, v) of the algorithm to a real value. This *potential function* should measure some sort of distance to the target state. We briefly discuss this potential function below. Proving that it yields the required multiplicative drift is the purpose of Lemma 18. The formal proof of the upper bound in Theorem 17 is rather technical and will be carried out in Sect. 7.3; while an overview of the main proof ideas is the focus of the discussion in the remainder of this section.

To simplify the notation below, for a given search point x and the target bit string z and the chosen metric d, we let $d_i = d(x_i, z_i)$ (for all $i \leq n$) be the distance vector of x to z. Thus, the goal is to reach a state in which the distance vector is $(0, \ldots, 0)$. We now want to define a potential function in dependence on (d, v) (where of course d is dependent on x) such that it is 0 when d is $(0, \ldots, 0)$ and strictly positive for any $x \neq (0, \ldots, 0)$. Furthermore, we easily see that there are two important ways to make progress, either by advancing in terms of fitness or by adjusting the velocity to a value that is more suitable to make progress in future iterations. This has to be reflected in the potential function. Our ultimate goal being the minimization of fitness, it is not difficult to see that some preference should be given to a progress in fitness. This can be achieved by multiplying the term accounting for the appropriateness of the velocity with some constant $c < 1$. We measure the appropriateness of the velocity as the maximum of the ratios $d_i/(2v_i)$ and $2v_i/d_i$, reflecting the fact that a velocity of $d_i/2$ is very well-suited for progress; smaller values give less progress, while larger values lead to a badly adjusted velocity in the next iteration (and very large values make progress in fitness impossible).

One problem in getting good drift is that velocities v_i just below $2d_i$ allow for jumping over the target while increasing the (already too large) velocity. We get around this problem by observing that it is equally likely that the large velocity is reduced because of a jump in the wrong direction, and then, while still larger than d_i, will still give a good improvement when overstepping the goal. We reflect this in the potential function by giving a penalty term of pd_i (for some suitable constant p) on any state (d, v) having a too large velocity.

To sum up this discussion we use as potential function the following map $g : [0 \ldots r-1]^n \times [1, \lfloor r/4 \rfloor]^n \to \mathbb{R}, (x, v) \mapsto \sum_{i=1}^{n} g_i(d_i, v_i)$ where $g_i(d_i, v_i) := 0$ for $d_i = 0$ and for $d_i \geq 1$

$$g_i(d_i, v_i) := d_i + \begin{cases} cd_i \max\{2v_i/d_i, d_i/(2v_i)\}, & \text{if } v_i \leq 2bd_i; \\ cd_i \max\{2v_i/d_i, d_i/(2v_i)\} + pd_i, & \text{otherwise} \end{cases} \tag{6}$$

and c, p are (small) constants specified below.

Summarizing all the conditions needed below, we require that that the constants a, b, c, p satisfy $1 < a \leq 2$, $1/2 < b \leq 0.9$, $2ab - b - a > 0$, $a + b > 2$, $a^2 b > 1$, $8abc + 2p + 4c/b \leq 1/16$, $p > 8c\left(\frac{a+b}{2} - 1\right)$, and $p > 4(a-1)c > 0$.

We can thus choose, for example, $a = 1.7$, $b = 0.9$, $p = 0.01$, and $c = 0.001$.

7.3 Proof of the Upper Bound in Theorem 17

The following lemma, together with the observation that the initial potential is of order at most nr^2 plugged into the multiplicative drift theorem (Theorem 1) proves the desired overall expected run time of $O(n \log(nr)) = O(n(\log n + \log r))$.

Lemma 18 *Let $d \neq (0, \ldots, 0)$ and $v \in [1, \lfloor r/4 \rfloor]^n$. Let (d', v') be the state of Algorithm 3 started in (d, v) after one iteration (i.e., after a possible update of x and v). The expected difference in potential satisfies*

$$E\left[g(d, v) - g(d', v') \mid d, v\right] \geq \frac{\delta}{n} g(d, v)$$

for some positive constant δ.

Proof Let d, d', v, and v' as in the statement of Lemma 18. Any fixed index i is chosen by Algorithm 3 for mutation with probability $1/n$; for all i, let A_i be the event that index i was chosen. We show that there is a constant δ such that, for all indices i with $d_i \neq 0$,

$$E\left[g_i(d_i, v_i) - g_i(d_i', v_i') \mid A_i\right] \geq \delta g_i(d_i, v_i),$$

thus proving the claim using $P(A_i) = 1/n$.

We regard several cases, depending on how d_i and v_i relate. Note that we do not consider updates into an infeasible area of the search space (which can happen in case of the interval metric). This would not change the argument, but it would involve a lot more cases, since the possibility of stepping into an infeasible region would have to be covered. Since such steps do not lead to any update at all, we only get the update from the only possible alternative, which in all cases gives the desired multiplicative drift.

Case 1 $v_i \leq d_i/8$.
First we observe that $\max\{2v_i/d_i, d_i/(2v_i)\} = d_i/(2v_i)$. The contribution of the i-th position to the current potential is thus

$$g_i(d_i, v_i) = d_i + cd_i^2/(2v_i).$$

With probability $1/2$ the algorithm decides to move in the right direction. In this case we make progress with respect to the fitness function and the velocity. That is, after the iteration we have $d_i' = d_i - \lfloor v_i \rfloor < d_i$ and $v_i' = \max\{av_i, r/4\} = av_i > v_i$. To see the second equality in the previous expression note that $av_i \leq 2d_i/8 \leq r/4$.

To bound the progress in the second component of g_i, we observe that

$$cd_i' \max\left\{2av_i/d_i', d_i'/(2av_i)\right\} = \max\left\{2cav_i, cd_i'^2/(2av_i)\right\} = cd_i'^2/(2av_i)$$

where the second equality follows from $2av_i \leq d_i/2 < d'_i$. We thus obtain that for this case the difference in potential is at least

$$g_i(d_i, v_i) - g_i\left(d'_i, v'_i\right) = d_i + cd_i^2/(2v_i) - d'_i - cd_i'^2/(2av_i) \geq \frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2av_i}. \quad (7)$$

With probability $1/2$ the algorithm decides to go the wrong direction. In case of the ring metric it is possible that (modulo r) $d'_i < d_i$ holds. In this case the offspring d' is accepted and the computations from above yield the same positive drift as above (since all we used about the new search point is that it improved). Otherwise, $d'_i > d_i$ holds and the new individual is thus discarded while the velocity v_i at position i is further decreased to $\max\{bv_i, 1\} \geq bv_i$. Hence, the difference in potential for this case is at least

$$g_i(d_i, v_i) - g_i(d_i, bv_i) = \frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2bv_i}. \quad (8)$$

Combining (7) and (8), we thus obtain that the expected difference in potential is at least

$$\begin{aligned}
\frac{1}{2}\left(\frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2av_i} + \frac{cd_i^2}{2v_i} - \frac{cd_i^2}{2bv_i}\right) &= \frac{cd_i^2}{2v_i}\left(\frac{2ab-b-a}{2ab}\right) \\
&= \left(\frac{2ab-b-a}{4ab}\right)\left(\frac{cd_i^2}{2v_i} + \frac{cd_i^2}{2v_i}\right) \geq \left(\frac{2ab-b-a}{4ab}\right)\left(4cd_i + \frac{cd_i^2}{2v_i}\right) \\
&\geq \left(\frac{2ab-b-a}{4ab}\right)\min\{4c, 1\}\left(d_i + \frac{cd_i^2}{2v_i}\right) \\
&= \left(\frac{2ab-b-a}{4ab}\right)\min\{4c, 1\} g_i(d_i, v_i),
\end{aligned}$$

where in the third step we have used the requirement that $v_i \leq d_i/8$.
Case 2 $d_i/8 < v_i \leq 2bd_i$.
Now we are in a range of velocity which is well-suited to make progress. In fact, every step towards the optimum decreases the distance to the optimum by at least the minimum of $\lfloor d_i/8 \rfloor$ (if v_i is close to $d_i/8$ and we hence do not overshoot the target) and $\lfloor (2-2b)d_i \rfloor$ (if $v_i = 2bd_i \geq d_i$ in which case we overshoot the target and the distance to it decreases from d_i to at most $\lfloor 2bd_i \rfloor - d_i$). In case of moving towards the target value, the change in the first term of g_i is thus at least

$$\min\{\lfloor d_i/8 \rfloor, \lfloor (2-2b)d_i \rfloor\} = \lfloor d_i/8 \rfloor,$$

using $b \leq 0.9$. However, note that the decrease is at least 1 (since v_i is at least 1). Furthermore, we have, for all $z \geq 8$, $z/16 \leq \lfloor z/8 \rfloor$. Thus, we always have a decrease of at least $d_i/16$.

We now compute the change in the second term of g_i. Regard first the case that $\max\{2v'_i/d'_i, d'_i/(2v'_i)\} = 2v'_i/d'_i$. In this case, we pessimistically assume that the previous contribution of the second term in $g_i(d_i, v_i)$ was zero. This contribution increases to at most

$$2cv'_i + pd'_i \le 2acv_i + pd'_i \le 2acv_i + pd_i \le (4abc + p)d_i. \tag{9}$$

If, on the other hand, $\max\{2v'_i/d'_i, d'_i/(2v'_i)\} = d'_i/(2v'_i)$ and the previous contribution of the second term in $g_i(d_i, v_i)$ was $cd_i^2/(2v_i)$ (note that this is in particular the case when the capping at $r/4$ is not in force; i.e., if $v' = av_i$, since in this case it holds that $v_i \le d'_i/(2a) \le d_i/(2a) < d_i/2$), then the contribution of this second term has been decreased to $c(d'_i)^2/(2av_i) \le cd_i^2/(2v_i)$. The change in contribution is thus positive in this case, and therefore in particular strictly larger than $-(4abc+p)d_i$. We finally need to regard the case that $\max\{2v'_i/d'_i, d'_i/(2v'_i)\} = d'_i/(2v'_i)$, $\max\{2v_i/d_i, d_i/(2v_i)\} = 2v_i/d_i$, and $av_i > \lfloor r/4 \rfloor$. In this case the contribution in the second term of g_i increases by at most

$$\frac{cd_i'^2}{2\lfloor r/4 \rfloor} \le \frac{cd_i'^2}{2av_i} \le \frac{cd_i^2}{2a(d_i/8)} \le \frac{4cd_i}{a} \le 4abcd_i,$$

where the last step follows from the condition $a^2b \ge 1$.

Summarizing this discussion, we see that in case of stepping towards the target the change in progress satisfies

$$g_i(d_i, v_i) - g_i\left(d'_i, v'_i\right) \ge d_i\left(1/16 - (4abc + p)\right), \tag{10}$$

which is positive by our conditions on c and p.

Let us now regard the case of stepping away from the optimum, which happens with probability 1/2. In case of the ring metric it is possible that the search point is nonetheless accepted. In this case we assume no progress in d_i and the contribution of the second part of the potential to drift we bound pessimistically with $(4abc + d)d_i$ from above just as in the computations for the step towards the target.

The more typical case is that the new search point is discarded and the velocity is decreased to $\max\{bv_i, 1\}$. Assume first that $\max\{bv_i, 1\} = bv_i$. Then,

$$g_i(d_i, v_i) - g_i\left(d'_i, v'_i\right) = \max\left\{2cv_i, \frac{cd_i^2}{2v_i}\right\} - \max\left\{2cbv_i, \frac{cd_i^2}{2bv_i}\right\}. \tag{11}$$

If $\max\{2cbv_i, cd_i^2/(2bv_i)\} = cd_i^2/(2bv_i)$, then the term in (11) is at least $-cd_i^2/(2bv_i) \ge -4cd_i/b$ by our condition $d_i/8 \le v_i$. Furthermore, if $\max\{2cbv_i, cd_i^2/(2bv_i)\} = 2cbv_i$, then (11) is strictly positive as can be seen by the following observation

$$\max\left\{2cv_i, \frac{cd_i^2}{2v_i}\right\} - 2cbv_i \ge 2cv_i - 2cbv_i > 0.$$

Putting everything together we thus obtain that for $d_i/8 \le v_i \le 2bd_i$

$$E\left[g_i(d_i, v_i) - g_i(d_i', v_i')\right] \le \frac{d_i}{2}\,(1/16 - 2(4abc + p) - 4c/b) \tag{12}$$

which is positive if $8abc + 2p + 4c/b \le 1/16$. Since $v_i = \Theta(d_i)$ this also shows that there is a positive constant δ such that $E\left[g_i(d_i, v_i) - g_i(d_i', v_i')\right] \ge \delta g_i(d_i, v_i)$.

We finally need to regard the case that $\max\{bv_i, 1\} = 1$. Intuitively, the cap can only make our situation better. This is formalized by the following computations. We need to bound

$$g_i(d_i, v_i) - g_i(d_i', v_i') = \max\left\{2cv_i, \frac{cd_i^2}{2v_i}\right\} - \max\left\{2c, \frac{cd_i^2}{2}\right\}. \tag{13}$$

As above we obtain positive drift for the case $\max\left\{2c, cd_i^2/2\right\} = 2c$ by observing that $\max\left\{2cv_i, \frac{cd_i^2}{2v_i}\right\} - 2c \ge 2cv_i - 2c \ge 0$ (using that $v_i \ge 1$). For the case $\max\left\{2c, cd_i^2/2\right\} = cd_i^2/2$ the term in (13) is at least $-cd_i^2/2 \ge -cd_i^2/(2bv_i) \ge -4cd_i/b$ as above. The same computation as above thus shows a positive multiplicative gain in g_i.

Case 3 $2bd_i < v_i < 2d_i$.

Under these conditions $g_i(d_i, v_i) = d_i + 2cv_i + pd_i$ holds.

As before, we first regard the case that the algorithm moves towards the target value. Since $b \ge 1/2$ it holds that $d_i \le 2bd_i < v_i$ and the target value is thus overstepped. However, due to the requirement $v_i < 2d_i$, the distance of the offspring is strictly smaller than the previous distance. The velocity is hence increased to $\min\{av_i, \lfloor r/4 \rfloor\} \le av_i$.

With probability 1/2 the algorithm does a step away from the goal. Since we have $d_i < v_i \le r/4$, we see that even if the ring-metric is in place such a step results in a rejected offspring and thus a reduced velocity $v_i' = \max\{bv_i, 1\}$. Regard first the case that $v_i' = bv_i$. Then, due to $bv_i < 2bd_i$, the penalty term pd_i is no longer applied and the resulting potential at component i is thus $g_i(d_i', v_i') = d_i + 2cbv_i$.

Ignoring any possible gains in d_i, we therefore obtain that the expected difference in the potential is at least

$$2cv_i\left(1 - \frac{a+b}{2}\right) + \frac{p}{2}d_i.$$

Note that $1 - (a+b)/2$ is negative, since we require $a + b > 2$. Using $v_i \le 2d_i$ we see that the drift is at least

$$4cd_i\left(1 - \frac{a+b}{2}\right) + \frac{p}{2}d_i = d_i\left(\frac{p}{2} - 4c\left(\frac{a+b}{2} - 1\right)\right).$$

Since $p > 8c\left(\frac{a+b}{2} - 1\right)$ this expression is positive. Furthermore, we have $g_i(d_i, v_i) = \Theta(d_i)$, yielding the desired multiplicative drift.

For $v'_i = 1$ we first observe that $v'_i = 1 \leq d_i \leq 2bd_i$ and the penalty term pd_i is thus not in force. Furthermore, we have $bd_i < bv_i \leq 1$ and thus $d_i \leq 1/b \leq 2$, showing that $\max\{2/d_i, d_i/2\} \leq \max\{2, 1\} = 2$. We obtain

$$E\left[g_i(d_i, v_i) - g(d'_i, v'_i)\right] \geq \frac{p}{2}d_i - cv_i(a-1) \geq \frac{p}{2}d_i - 2(a-1)cd_i,$$

which is positive for $p/2 - 2(a-1)c > 0$.
Case 4 $v_i = 2d_i$.
Just as in the previous case, steps away from the target are not accepted, as can be seen by observing that $d_i < v_i \leq r/4$. Thus, regardless of whether or not we move towards or away from the target, the fitness does not decrease; therefore, the velocity is decreased to bv_i (note that $v_i \geq 2$ and hence $bv_i \geq 1$). The previous contribution of the i-th component to $g(x)$ being $d_i + 2cv_i + pd_i = d_i(1 + 4c + p)$, and the new potential at the i-th component being $d_i(1 + 4bc)$, we obtain

$$E\left[g_i(d_i, v_i) - g(d'_i, v'_i)\right] = d_i(4c + p - 4bc),$$

which is strictly positive and linear in $g_i(d_i, v_i)$.
Case 5 $2d_i < v_i$.
Steps towards the optimum are now not accepted, since they overstep the optimum by too much. Steps away from the optimum are also not accepted since again we have $d_i < v_i \leq r/4$. Therefore, we always decrease the velocity to $\max\{bv_i, 1\} = bv_i$ (note that $v_i > 2$ and thus $bv_i > 1$) and the gain in potential is

$$2cv_i + pd_i - [2cbv_i + pd_i] = 2cv_i(1-b) > 4(1-b)cd_i,$$

showing that we have multiplicative drift as desired. □

8 Discussion of Our Results

While many analyses of randomized search heuristics focus on the behavior of the algorithm in dependence on a large and growing dimension, we additionally considered a growing size of the search space in each dimension. We considered RLS and the (1+1) EA with different mutation strengths and proved asymptotically tight optimization times for a variety of OneMax-type test functions over an alphabet of size r. We proved that both using large changes (change to uniformly chosen different value) or very local changes (change value by ± 1) leads to relatively slow (linear in r) optimization times of $\Theta(rn \log n)$ and $\Theta(n(r + \log n))$, respectively.

We then considered a variable step size operator which allows for both large and small steps with reasonable probability; this leads to an optimization time of $\Theta(n \log r(\log n + \log r))$. Note that this bound, while polylogarithmic in r, is worse than the bound of $\Theta(n(r+\log n))$ for the ± 1 operator when r is asymptotically smaller than $\log n \log\log n$. This shows that there is no uniform superior mutation operator among the three proposed static ones. It remains open whether there is yet another "natural" static mutation operator achieving a better runtime than each of these two.

We have also shown that a dynamic choice of the mutation strength can bring the dependence on r further down to logarithmic. In fact, already a simple success-based update rule achieves this optimal dependence. We have investigated this dynamic parameter choice only for a variant of RLS. However, already its analysis required a rather intricate drift-argument, with many different cases to consider and penalty terms for resolving situations which would otherwise allow for search points with negative drift. Extending our results to the case of the $(1+1)$ EA might thus be a very challenging task, pushing the limits of drift theory.

Note that we chose a specific step size adaptation scheme which guarantees optimal run time. It would also be interesting to investigate other adaptation schemes. For example, the step size, in each iteration, could be drawn from a distribution (just as in one of the operators presented in [13]), and the parameters of this distribution are adapted.

Another issue with step sizes is that infeasible areas of the search space might be reached (in our setting this can happen if we use the interval metric). The issue of boundary handling is a known problem, and our boundary handling technique is by no means the only way for dealing with it. We believe that our choice is natural and leads to a "fair" treatment of all parts of the search space, and it leads to an optimal run time for our setting. It might be interesting to see whether there are other settings where a different boundary handling is more natural, or gives better run time.

Acknowledgements This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Programme Gaspard Monge en Optimisation et Recherche Opérationnelle.

References

1. Auger, A., Doerr, B.: Theory of Randomized Search Heuristics. World Scientific, Singapore (2011)
2. Auger, A., Hansen, N.: Linear convergence on positively homogeneous functions of a comparison based step-size adaptive randomized search: the (1+1) ES with generalized one-fifth success rule. CoRR (2013). arXiv:1310.8397
3. Badkobeh, G., Lehre, P.K., Sudholt, D.: Unbiased black-box complexity of parallel search. In: Proceedings of Parallel Problem Solving from Nature (PPSN'14), Lecture Notes in Computer Science, vol. 8672, pp. 892–901. Springer (2014)
4. Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In: Proceedings of Parallel Problem Solving from Nature (PPSN'10), Lecture Notes in Computer Science, vol. 6238, pp. 1–10. Springer (2010)
5. Buzdalov, M., Doerr, B.: Runtime analysis of the $(1+(\lambda,\lambda))$ genetic algorithm on random satisfiable 3-CNF formulas. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'17). ACM (2017)
6. Dang, D., Lehre, P.K.: Self-adaptation of mutation rates in non-elitist populations. In: Proceedings of Parallel Problem Solving from Nature (PPSN'16), Lecture Notes in Computer Science, vol. 9921, pp. 803–813. Springer (2016)
7. Dietzfelbinger, M., Rowe, J.E., Wegener, I., Woelfel, P.: Tight bounds for blind search on the integers and the reals. Comb. Probab. Comput. **19**, 711–728 (2010)
8. Doerr, B.: Analyzing randomized search heuristics: tools from probability theory. In: Auger, A., Doerr, B. (eds.) Theory of Randomized Search Heuristics, pp. 1–20. World Scientific Publishing, Singapore (2011)
9. Doerr, B., Doerr, C.: The impact of random initialization on the runtime of randomized search heuristics. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'14), pp. 1375–1382. ACM (2014)

10. Doerr, B., Doerr, C.: Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'15), pp. 1335–1342. ACM (2015)
11. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theor. Comput. Sci. **567**, 87–104 (2015)
12. Doerr, B., Doerr, C., Kötzing, T.: Provably optimal self-adjusting step sizes for multi-valued decision variables. In: Proceedings of Parallel Problem Solving from Nature (PPSN'16), Lecture Notes in Computer Science, vol. 9921, pp. 782–791. Springer (2016)
13. Doerr, B., Doerr, C., Kötzing, T.: The right mutation strength for multi-valued decision variables. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'16), pp. 1115–1122. ACM (2016)
14. Doerr, B., Doerr, C., Yang, J.: k-bit mutation with self-adjusting k outperforms standard bit mutation. In: Proceedings of Parallel Problem Solving from Nature (PPSN'16), Lecture Notes in Computer Science, vol. 9921, pp. 824–834. Springer (2016)
15. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'16), pp. 1123–1130. ACM (2016)
16. Doerr, B., Fouz, M., Witt, C.: Sharp bounds by probability-generating functions and variable drift. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'11), pp. 2083–2090. ACM (2011)
17. Doerr, B., Gießen, C., Witt, C., Yang, J.: The (1+λ) evolutionary algorithm with self-adjusting mutation rate. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'17). ACM (2017)
18. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. Algorithmica **65**, 224–250 (2013)
19. Doerr, B., Johannsen, D.: Adjacency list matchings: an ideal genotype for cycle covers. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'07), pp. 1203–1210. ACM (2007)
20. Doerr, B., Johannsen, D., Schmidt, M.: Runtime analysis of the (1+1) evolutionary algorithm on strings over finite alphabets. In: Proceedings of Foundations of Genetic Algorithms (FOGA'11), pp. 119–126. ACM (2011)
21. Doerr, B., Johannsen, D., Winzen, C.: Multiplicative drift analysis. Algorithmica **64**, 673–697 (2012)
22. Doerr, B., Pohl, S.: Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'12), pp. 1317–1324. ACM (2012)
23. Droste, S., Jansen, T., Wegener, I.: Upper and lower bounds for randomized search heuristics in black-box optimization. Theory Comput. Syst. **39**, 525–544 (2006)
24. Eiben, A.E., Hinterding, R., Michalewicz, Z.: Parameter control in evolutionary algorithms. IEEE Trans. Evolut. Comput. **3**, 124–141 (1999)
25. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2003)
26. Gunia, C.: On the analysis of the approximation capability of simple evolutionary algorithms for scheduling problems. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'05), pp. 571–578. ACM (2005)
27. Hansen, N., Gawelczyk, A., Ostermeier, A.: Sizing the population with respect to the local progress in (1,λ)-evolution strategies—a theoretical analysis. In: Proceedings of IEEE Congress on Evolutionary Computation (CEC'95), pp. 80–85. IEEE (1995)
28. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artif. Intell. **127**, 57–85 (2001)
29. Jägersküpper, J.: Rigorous runtime analysis of the (1+1) ES: 1/5-rule and ellipsoidal fitness landscapes. In: Proceedings of Foundations of Genetic Algorithms (FOGA'05), Lecture Notes in Computer Science, vol. 3469, pp. 260–281. Springer (2005)
30. Jägersküpper, J.: Oblivious randomized direct search for real-parameter optimization. In: Proceedings of European Symposium on Algorithms (ESA), Lecture Notes in Computer Science, vol. 5193, pp. 553–564. Springer (2008)
31. Jansen, T.: Analyzing Evolutionary Algorithms—The Computer Science Perspective. Springer, Berlin (2013)
32. Jansen, T., Wegener, I.: On the analysis of a dynamic evolutionary algorithm. J. Discrete Algorithms **4**, 181–199 (2006)
33. Johannsen, D.: Random combinatorial structures and randomized search heuristics. Ph.D. thesis, Saarland University. http://scidok.sulb.uni-saarland.de/volltexte/2011/3529/ (2010)

34. Karafotias, G., Hoogendoorn, M., Eiben, A.: Parameter control in evolutionary algorithms: trends and challenges. IEEE Trans. Evolut. Comput. **19**, 167–187 (2015)
35. Kötzing, T., Lissovoi, A., Witt, C.: (1+1) EA on generalized dynamic OneMax. In: Proceedings of Foundations of Genetic Algorithms (FOGA'15), pp. 40–51. ACM (2015)
36. Lässig, J., Sudholt, D.: Adaptive population models for offspring populations and parallel evolutionary algorithms. In: Proceedings of Foundations of Genetic Algorithms (FOGA'11), pp. 181–192. ACM (2011)
37. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**, 623–642 (2012)
38. Lissovoi, A., Witt, C.: MMAS vs. population-based EA on a family of dynamic fitness functions. In: Proceedings of Genetic and Evolutionary Computation Conference (GECCO'14), pp. 1399–1406. ACM (2014)
39. Mitavskiy, B., Rowe, J., Cannings, C.: Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. Int. J. Intell. Comput. Cybern. **2**, 243–284 (2009)
40. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization—Algorithms and Their Computational Complexity. Springer, Berlin (2010)
41. Oliveto, P.S., Lehre, P.K., Neumann, F.: Theoretical analysis of rank-based mutation-combining exploration and exploitation. In: Proceedings of Congress on Evolutionary Computation (CEC'09), pp. 1455–1462. IEEE (2009)
42. Rothlauf, F.: Representations for Genetic and Evolutionary Algorithms, 2nd edn. Springer, Berlin (2006)
43. Rudolph, G.: An evolutionary algorithm for integer programming. In: Proceedings of Parallel Problem Solving from Nature (PPSN'94), pp. 139–148. Springer (1994)
44. Scharnow, J., Tinnefeld, K., Wegener, I.: The analysis of evolutionary algorithms on sorting and shortest paths problems. J. Math. Model. Algorithms **3**, 349–366 (2004)
45. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**, 294–318 (2013)
46. Zarges, C.: Rigorous runtime analysis of inversely fitness proportional mutation rates. In: Proceedings of Parallel Problem Solving from Nature (PPSN'08), Lecture Notes in Computer Science, vol. 5199, pp. 112–122. Springer (2008)
47. Zarges, C.: On the utility of the population size for inversely fitness proportional mutation rates. In: Proceedings of Foundations of Genetic Algorithms (FOGA'09), pp. 39–46. ACM (2009)