The Right Mutation Strength for Multi-Valued Decision Variables

Benjamin Doerr
Laboratoire d'Informatique (LIX)
École Polytechnique
Paris-Saclay, France

Carola Doerr
CNRS & LIP6
Univ. Pierre et Marie Curie
Paris, France

Timo Kötzing
Hasso-Plattner-Institut
Potsdam, Germany

ABSTRACT

The most common representation in evolutionary computation are bit strings. This is ideal to model binary decision variables, but less useful for variables taking more values. With very little theoretical work existing on how to use evolutionary algorithms for such optimization problems, we study the run time of simple evolutionary algorithms on some OneMax-like functions defined over $\Omega = \{0, 1, \dots, r-1\}^n$. More precisely, we regard a variety of problem classes requesting the component-wise minimization of the distance to an unknown target vector $z \in \Omega$.

For such problems we see a crucial difference in how we extend the standard-bit mutation operator to these multi-valued domains. While it is natural to select each position of the solution vector to be changed 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 either $+1$ or -1 (random choice), the optimization time reduces to $\Theta(nr+n \log n)$. If we use a random mutation strength $i \in \{0, 1, \dots, r-1\}^n$ with probability inversely proportional to i and change the selected position by either $+i$ or $-i$ (random choice), then the optimization time becomes $\Theta(n \log(r)(\log(n) + \log(r)))$, bringing down the dependence on r from linear to polylogarithmic.

One of our results depends on a new variant of the lower bounding multiplicative drift theorem.

Categories and Subject Descriptors

F.2.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity—*Nonnumerical Algorithms and Problems*

Keywords

Run Time Analysis; Theory; Large Alphabet; Mutation

GECCO '16, July 20 - 24, 2016, Denver, CO, USA

ISBN 978-1-4503-4206-3/16/07…$15.00

DOI: http://dx.doi.org/10.1145/2908812.2908891

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 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) or to re-model the problem using a bit string representation. For an example for the latter, see, e.g., [8], where the Eulerian cycle problem (asking for a permutation of the edges) was re-modeled as a matching problem. In general, such a re-modeling may not lead to an efficient or a natural approach, and it may be better to work with a representation different from bit strings. The traveling salesman problem is an example for such a situation.

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 [13]. 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 [24] to design evolutionary algorithms for shortest-path problems.

An alternative representation would be to code each value in $\log r$ bits, leading to a search space of $\{0,1\}^{n \log r}$. However, this representation has the weakness that search points with similar fitness can be vastly different (the bit representations $10\dots0$ and $01\dots1$ code almost the same value,

but are complementary); this trap-like behavior can lead to a very poor performance on some OneMax functions (see Section 1.2 for a formal defintion).

1.1 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 $i \in [1..n] := \{1, \dots, n\}$, that is, in the factors $[0..r-1]$, then the natural analogue of the standard-bit mutation operator is to select each component $i \in [1..n]$ independently and mutate the selected components by changing the current value to a random other value in $[0..r-1]$. This operator was used in [13,24] as well as in the theoretical works [9,12].

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 the principle of standard-bit mutation that each component $i \in [1..n]$ is changed independently with probability $1/n$, so the only point of discussion is how such an elementary change looks like. 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 Section 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 $[1..r-1]$. 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 was used in [2] and is called *harmonic distribution*, aiming at overcoming the two individual weaknesses of the two operators discussed before and, as we shall see, this does indeed work.

1.2 Run time Analysis of Multi-Valued OneMax Functions

To gain a more 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., [16]. 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., [22]).

On bit strings, the classic OneMax test function is defined by $\textsc{Om} : \{0,1\}^n \to \mathbb{R}; (x_1, \dots, x_n) \mapsto \sum_{i=1}^n x_i$. 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 $\textsc{Om} : \{0, 1, \dots, r-1\}^n \to \mathbb{R}; x \mapsto \sum_{i=1}^n x_i$, 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 $\textsc{Om}(x) = H(x, (0, \dots, 0))$, where $H(x,y) := |\{i \in [1..n] \mid x_i \neq y_i\}|$ denotes the Hamming distance between two bit strings x and y. Defining $f_z : \{0,1\}^n \to \mathbb{R}; x \mapsto H(x,z)$ 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 $f_z : \{0, 1, \dots, r-1\}^n \mapsto \mathbb{R}; x \mapsto \sum_{i=1}^n |x_i - z_i|$ for all $z \in \{0, 1, \dots, r-1\}^n$, 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.

As evolutionary algorithm to optimize these test functions, we study the (1+1) evolutionary algorithm (EA). This is arguably the most simple evolutionary algorithm, however, many results that could first only be shown for the $(1+1)$ EA could later be extended to more complicated algorithms, making it an ideal instrument for a first study of a new subject. Naturally, to study mutation operators we prefer mutation-based EAs. 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 evolutionary algorithm needs to find the optimal solution. This *optimization time* today is one of the most accepted performance measures for evolutionary algorithms.

1.3 Previous Works and Our Results

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 rescaling things into, say, $\{0, \frac{1}{r}, \frac{2}{r}, \dots, 1\}^n$, 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 Jägerskküpper (see [15] and the references therein), which regards continuous optimization with an a-priori fixed target preci-

sion. 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., [24]); 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 [9] 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 [12], 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 [18]. 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 [20] 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 three different mutation strengths. For the uniform mutation strength, we show a tight and precise $(1+o(1))e(r-1)n\ln(n)$ run time estimate for all values of r (Section 4). For the cautious ± 1 mutation strength, the run time becomes $\Theta(n(r+\log n))$, that is, still (mostly) linear in r (Section 5). The harmonic mutation strength overcomes this slowness 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 bound (Section 6).

All analyses rely on drift methods, for the lower bound for the case of uniform mutation strength we prove a variant of the multiplicative drift lower bound theorem [25] that does not need the restriction that the process cannot go back to inferior search points (see Section 4.2.2).

For reasons of space, several proofs had to be omitted in this extended abstract. They can be found in [5].

2. ALGORITHMS AND PROBLEMS

In this section we define the algorithms and problems considered in this paper. We let $[r] := \{0,1,\dots,r-1\}$ and $[1..r] := \{1,2,\dots,r\}$. 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 $\Omega = [r]^n$.

We define the following two metrics on $[r]$, 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 $a, b \in [r]$,

$$\begin{aligned} d_{\text{int}}(a,b) &= |b-a|; \\ d_{\text{ring}}(a,b) &= \min\{|b-a|, |b-a+r|, |b-a-r|\}. \end{aligned}$$

We consider different *step operators* $v : [r] \to [r]$ (possibly randomized). These step operators will later decide the update of a mutation in a given component. Thus we call, for any given $x \in [r]$, $d(x, v(x))$ the *mutation strength*. We consider the following step operators.
(1) The *uniform step* operator chooses a different element from $[r]$ uniformly at random; thus we speak of a *uniform mutation strength*.
(2) The ± 1 operator chooses to either add or subtract 1, each with probability $1/2$; this operator has a mutation strength of 1.
(3) The *Harmonic* operator makes a jump of size $j \in [r]$ 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 and discard this mutation as infeasible. Note that this somewhat arbitrary choice does not impact the results in this paper.

We consider the algorithms RLS and $(1+1)$ EA as given by Algorithms 1 and 2. Both algorithms sample an initial search point from $[r]^n$ 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 $f(y) \le f(x)$. 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).

Note that the algorithms with the considered topologies are unbiased in the general sense of [23] (introduced for $\{0,1\}^n$ by Lehre and Witt [19] and made specific for several combinatorial search spaces in [11]).

Let d be either the interval- or the ring-metric and let $z \in [r]^n$. We can define a straightforward generalization of the OneMax fitness function as $\sum_{i=1}^n d(x_i, z_i)$. 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*.

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

Initialization: Sample $x \in [r]^n$ uniformly at random and query $f(x)$;
Optimization: **for** $t = 1, 2, 3, \ldots$ **do**
  Choose $i \leq n$ uniformly at random;
  for $j = 1, \ldots, n$ **do**
    if $j = i$ **then** $y_j \leftarrow v(x_j)$;
    else $y_j \leftarrow x_j$;
  Evaluate $f(y)$;
  if $f(y) \leq f(x)$ **then** $x \leftarrow y$;

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

Initialization: Sample $x \in [r]^n$ uniformly at random and query $f(x)$;
Optimization: **for** $t = 1, 2, 3, \ldots$ **do**
  for $i = 1, \ldots, n$ **do**
    With probability $1/n$ set $y_i \leftarrow v(x_i)$ and set $y_i \leftarrow x_i$ otherwise;
  Evaluate $f(y)$;
  if $f(y) \leq f(x)$ **then** $x \leftarrow y$;

3. DRIFT ANALYSIS

A central tool in 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. It was first used in evolutionary computation by He and Yao [14] and is now, after a large number of subsequent works, probably the most powerful tool in run time analysis. We briefly introduce the necessary tools.

We phrase the following results in the language that we have some random process, either in the real numbers or in some other set Ω, but then equipped with a potential function $g : \Omega \to \mathbb{R}$. 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 [10]. A more direct proof of this results, that also gives large deviation bounds, was later given in [7].

Theorem 1 (from [10]). *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 [25], 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 7) that the first restriction can be removed under not too strong additional assumptions.

Theorem 2 (from [25]). *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 [17] can lead to very good results. Another version of a variable drift theorem can be found in [21, Lemma 8.2].

Theorem 3 (from [17]). *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*

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

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 Section 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 $g : [r]^n \to \mathbb{R}; x \mapsto H(x, z) = |\{i \in [1..n] \mid x_i \neq z_i\}|$, 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 4. *Let f be any r-valued OneMax function with target $z \in [r]^n$. 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 $x \in [r]^n$ we have $E[T | x_0 = x] = n(r - 1)H_{H(x,z)}$, where, for

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

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 5. *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 $x = (r, 0, \dots, 0)$ and $y = (1, 0, \dots, 0)$ 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

$$(1-1/n)^{n-2}(n-1)(1/n^2)(1+2+\cdots+(r-2))/(r-1)^2 = (1 \pm o(1))\frac{r-2}{2e(r-1)n},$$

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. 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 6. *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 $[r]$ that were described in Section 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

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

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 $[1..n]$. 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 $x_i \in [r]$ with $d(x_i, z_i) = k$, then for each $j \in [0..k-1]$ there is at least one value $y_i \in [r]$ such that $d(y_i, z_i) = j$. This shows (1).

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 [17] variable drift theorem (Theorem 3). Let $S = [0..(r-1)n]$. 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

In this section, we write $(q)_+ := \max\{q, 0\}$ for any $q \in \mathbb{R}$. 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 have to overcome one difficulty, though, and this is 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 7 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 $s \le g(X^{(t)})$. 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 8). 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 [25]. Previous works, e.g., [25], 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 8 is a convenient way to overcome these difficulties.

We start by proving our new drift results, then compute that the Hamming distance to the optimum satisfies the assumptions of our drift results, and finally state and prove the precise lower bound.

Theorem 7. (MULTIPLICATIVE DRIFT, LOWER BOUND, NON-DECREASING PROCESS) *Let $X^{(t)}, t = 0, 1, \dots$ 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 $s_{\text{aim}} \ge 1$. Let $T := \min\{t \ge 0 \mid g(X^{(t)}) \le s_{\text{aim}}\}$ be the random variable describing the first point in time for which $g(X^{(t)}) \le s_{\text{aim}}$.*

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

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

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

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 $Y^{(t)} := \min\{g(X^{(\tau)}) \mid \tau \in [0..t]\}$.

Corollary 8. *Assume that the assumptions of Theorem 7 are satisfied, however with δ replaced by $\delta(s)$ for some function $\delta : S \to (0, 1]$. Then for any $s_{\text{aim}} < \tilde{s}_0 \le s_0$, we have*

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

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

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 $\omega \in \Omega$, $s_{\text{aim}} < s \le \tilde{g}(\omega)$, and t be such that $\Pr[X^{(t)} = \omega] > 0$. Then

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

by the assumptions of Theorem 7 and $s \le \tilde{s}_0$. Similarly,

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

Hence we may apply Theorem 7 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. □

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

Theorem 9. *For any r-valued* ONEMAX *function, the $(1+1)$ EA with uniform mutation strength has an expected optimization time of*

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

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. We give the full proof below since it is similar to the analysis of the $(1+1)$ EA equipped with the ±1 variation operator. The proof makes 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 10. *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 $x \in \Omega = [r]^n$, let

$$g(x) := \sum_{i=1}^{n} (w^{d(z_i,x_i)} - 1), \tag{2}$$

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 $i \in [1..n]$, and denoting by $O := \{i \in [1..n] \mid x_i = z_i\}$ the set of already optimized bits,

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

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 $\ln(nw^r) / \left(\tfrac{1}{2n}(1 - \tfrac{1}{w})\right) = O(n(\log(n) + r))$, as desired. □

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

Theorem 11. *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.*

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 $j \in [1..r-1]$ 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 12. *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-values 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.

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 [2], where the following result is presented, which could also be used to derive the run time bound of Theorem 12.

Theorem 13 ([2]). *Let a random process on $A = \{0, \dots, r\}$ be given, representing the movement of a token. Fix a probability distribution of step sizes D over $\{1, \dots, r\}$. 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 $\min_D(E[T_D]) = \Theta((\log r)^2)$.*

While our processes have a slightly different behavior (including the possibility to overshoot the goal), we believe that these differences only lead to to differences in the constants of the optimization time. Thus, the above theorem indicates that the Harmonic distribution is an optimal choice and cannot be improved.

7. CONCLUSION

While many analyses of randomized search heuristics focus on the behavior of the algorithm in dependence of a large and growing dimension, we additionally considered a growing size of the search space in each dimension. We considered the $(1 + 1)$ EA with three 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 (essentially 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 operators.

Acknowledgments. This research benefited from the support of the "FMJH Program Gaspard Monge in optimization and operation research", and from the support to this program from EDF (Électricité de France).

8. REFERENCES

[1] A. Auger and B. Doerr. *Theory of Randomized Search Heuristics*. World Scientific, 2011.

[2] M. Dietzfelbinger, J. E. Rowe, I. Wegener, and P. Woelfel. Tight bounds for blind search on the integers and the reals. *Combinatorics, Probability & Computing*, 19:711–728, 2010.

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

[4] B. Doerr and C. Doerr. The impact of random initialization on the runtime of randomized search heuristics. In *GECCO*, pages 1375–1382. ACM, 2014.

[5] B. Doerr, C. Doerr, and T. Kötzing. The right mutation strength for multi-valued decision variables. *CoRR*, abs/1604.03277, 2016.

[6] B. Doerr, M. Fouz, and C. Witt. Sharp bounds by probability-generating functions and variable drift. In *GECCO*, pages 2083–2090. ACM, 2011.

[7] B. Doerr and L. A. Goldberg. Adaptive drift analysis. *Algorithmica*, 65:224–250, 2013.

[8] B. Doerr and D. Johannsen. Adjacency list matchings: an ideal genotype for cycle covers. In *GECCO*, pages 1203–1210. ACM, 2007.

[9] B. Doerr, D. Johannsen, and M. Schmidt. Runtime analysis of the (1+1) evolutionary algorithm on strings over finite alphabets. In *FOGA*, pages 119–126. ACM, 2011.

[10] B. Doerr, D. Johannsen, and C. Winzen. Multiplicative drift analysis. *Algorithmica*, 64:673–697, 2012.

[11] B. Doerr, T. Kötzing, J. Lengler, and C. Winzen. Black-box complexities of combinatorial problems. *Theoretical Computer Science*, 471:84–106, 2013.

[12] B. Doerr and S. Pohl. Run-time analysis of the (1+1) evolutionary algorithm optimizing linear functions over a finite alphabet. In *GECCO*, pages 1317–1324. ACM, 2012.

[13] C. Gunia. On the analysis of the approximation capability of simple evolutionary algorithms for scheduling problems. In *GECCO*, pages 571–578. Jahn, 2005.

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

[15] J. Jägersküpper. Oblivious randomized direct search for real-parameter optimization. In *ESA*, pages 553–564. Springer, 2008.

[16] T. Jansen. *Analyzing Evolutionary Algorithms—The Computer Science Perspective*. Springer, 2013.

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

[18] T. Kötzing, A. Lissovoi, and C. Witt. (1+1) EA on generalized dynamic OneMax. In *FOGA*, pages 40–51. ACM, 2015.

[19] P. K. Lehre and C. Witt. Black-box search by unbiased variation. *Algorithmica*, 64:623–642, 2012.

[20] A. Lissovoi and C. Witt. MMAS vs. population-based EA on a family of dynamic fitness functions. In *GECCO*, pages 1399–1406. ACM, 2014.

[21] B. Mitavskiy, J. Rowe, and C. Cannings. Theoretical analysis of local search strategies to optimize network communication subject to preserving the total number of links. *International Journal of Intelligent Computing and Cybernetics*, 2:243–284, 2009.

[22] F. Neumann and C. Witt. *Bioinspired Computation in Combinatorial Optimization – Algorithms and Their Computational Complexity*. Springer, 2010.

[23] J. Rowe and M. Vose. Unbiased black box search algorithms. In *GECCO*, pages 2035–2042. ACM, 2011.

[24] J. Scharnow, K. Tinnefeld, and I. Wegener. The analysis of evolutionary algorithms on sorting and shortest paths problems. *J. Math. Model. Algorithms*, 3:349–366, 2004.

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