Comparing Global and Local Mutations on Bit Strings[*]

Benjamin Doerr
Max-Planck-Institut für Informatik
Campus E 1 4
Saarbrücken, Germany

Thomas Jansen
Technische Universität Dortmund
Otto-Hahn-Straße 14
Dortmund, Germany

Christian Klein
Max-Planck-Institut für Informatik
Campus E 1 4
Saarbrücken, Germany

ABSTRACT

Evolutionary algorithms operating on bit strings usually employ a global mutation where each bit is flipped independently with some mutation probability. Most often the mutation probability is set fixed in a way that on average exactly one bit is flipped in a mutation. A seemingly very similar concept is a local one realized by an operator that flips exactly one bit chosen uniformly at random.

Most known results indicate that the global approach leads to run-times at least as good as the local approach. The draw-back is that the global approach is much harder to analyze. It would therefore be highly useful to derive general principles of when and how results for the local operator extend to the global ones.

In this paper, we show that there is little hope for such general principles, even under very favorable conditions. We show that there is a fitness function such that the local operator from each initial search point finds the optimum in small polynomial time, whereas the global operator for almost all initial search points needs a weakly exponential time.

Categories and Subject Descriptors

F.2 [**Theory of Computation**]: Analysis of Algorithms and Problem Complexity

General Terms

Theory, Algorithms

Keywords

Evolutionary Computation, Randomized Local Search, Mutation, Analysis

[*]This work benefited from being presented and discussed at the Dagstuhl seminar 08051 on Theory of Evolutionary Algorithms.

1. INTRODUCTION

Evolutionary algorithms (EAs) are typically described as robust general problem solvers. They are able to perform a global search different from gradient-descent methods or hill-climbers, which easily are trapped in local optima.

It is in fact easy to prove that evolutionary algorithms find a global optimum with probability converging to 1 with time if they make use of a positive mutation operator, i. e., a mutation that changes any point in the search space to any other point in the search space with positive probability. If an EA operates on bit strings of fixed length n, the most commonly used mutation operator is standard bit mutation. With standard bit mutation, each bit is flipped independently with a fixed mutation probability p_m. The most recommended choice for the mutation probability is $p_m = 1/n$.

Clearly, with mutation probability $p_m = 1/n$, on average exactly 1 bit is flipped in each mutation. Therefore, it seems to be a small change to replace standard bit mutation by a local mutation operator that flips exactly one bit chosen uniformly at random. However, with such a local mutation operator the EA may now get stuck in a local optimum, and consequently, the probability to finally reach the global optimum might no longer converge to 1 with time. In addition, most results indicate that the local operator in case of convergence leads to similar run-times as the global one. Therefore, one might be tempted to believe that the global operator generally is superior to the local one. Since typically rigorous analyses for the global operator are much harder than for the local one, general results of how a good optimization behavior of the local operator extends to the global one would be highly desirable.

When analyzing such general phenomena of evolutionary algorithms one often considers particularly simple evolutionary algorithms to facilitate a rigorous analysis. The probably most simple example is the well-known (1+1) EA. It uses a population of size only 1, produces only 1 offspring using standard bit mutation and a plus-selection. Thus, the parent x is replaced by its offspring y if and only if $f(y) \geq f(x)$ holds (assuming that we want to maximize the fitness function f). If we replace standard bit mutation by a local mutation operator that flips exactly one bit chosen uniformly at random, we obtain an algorithm that is well known as randomized local search (RLS). Since RLS is a hill-climber and no evolutionary algorithm, the (1+1) EA is right on the borderline and a comparison of RLS and the (1+1) EA is a comparison between an evolutionary algorithm and a simpler search heuristic. This is one motivation for comparing the performance of these two algorithms in a rigorous way.

As indicated above, in many cases the analysis of RLS is much simpler than that of the (1+1) EA. For example, for linear functions an upper bound of $O(n \log n)$ for the expected optimization time of RLS follows as a direct consequence of the coupon collector's theorem [8], simply because it suffices that each bit was touched once by the algorithm.

For the (1+1) EA, things are more complicated. The reason is that mutations involving more than one bit may result in some bits being flipped "in the wrong direction". Hence to prove the $O(n \log n)$ bound, which holds as well, much more work is necessary. Currently, there are two proofs for this results, a rather complicated analysis making use of a potential function [4] and one using deep methods like drift analysis [5]. Hence a simple results telling that (under certain conditions) results for RLS carry over to the (1+1) EA would be highly desirable. Even a by far less precise statement describing for which functions a polynomial upper bound on the expected optimization time for RLS implies some (other) polynomial upper bound on the expected optimization time for the (1+1) EA would be of interest.

In this paper, however, we show that such a characterization is unlikely to exist. Even under relatively strong conditions, namely that RLS finds the optimum from any initial search point in polynomial time, it can happen that the (1+1) EA needs weakly exponential time to find the optimum for almost all initial search points. This shows that the existence of more-bit flips can significantly put the EA behind.

In the next section we give precise formal definitions of the (1+1) EA and RLS, describe our analytical model, and define useful tools. First simple results showing extreme performance differences are presented and discussed in Section 3. We discuss what intuition follows from these examples and prove this intuition wrong in Section 4. Finally, we conclude and discuss directions of possible future research in Section 5.

2. ALGORITHMS AND ANALYTICAL FRAMEWORK

We begin the formal description of our objects of study with the definition of the two algorithms under consideration, randomized local search (RLS) and the (1+1) evolutionary algorithm ((1+1) EA). We describe both algorithms without stopping criterion since we are interested in the first hitting time of a global optimum. By y_i we denote the i-th bit in a bit-string $y \in \{0,1\}^n$.

Algorithm 1 ((1+1) EA).

```
1. Initialization
     Choose $x^{(1)} \in \{0,1\}^n$ uniformly at random. Set $t := 1$.
2. Mutation
     Set $y := x^{(t)}$. Independently for each $i \in \{1,2,\dots,n\}$,
     with probability $1/n$ set $y_i := 1 - y_i$.
3. Selection
     If $f(y) \geq f(x^{(t)})$ Then $x^{(t+1)} := y$ Else $x^{(t+1)} := x^{(t)}$.
4. Set $t := t + 1$. Continue at line 2.
```

Algorithm 2 (Randomized Local Search (RLS)).

```
1. Initialization
     Choose $x^{(1)} \in \{0,1\}^n$ uniformly at random. Set $t := 1$.
2. Random Selection from Neighborhood
     Choose $y \in \left\{x \mid H\left(x, x^{(t)}\right) = 1\right\}$ uniformly at random.
3. Hill Climbing
     If $f(y) \geq f(x^{(t)})$ Then $x^{(t+1)} := y$ Else $x^{(t+1)} := x^{(t)}$.
4. Set $t := t + 1$. Continue at line 2.
```

Clearly, the (1+1) EA (Algorithm 1) and RLS (Algorithm 2) differ only in the way the next potential search point is chosen in line 2. As we shall discuss in the following, this can make a huge difference in performance. As usual, we measure the performance of our algorithms by means of the so-called optimization time.

Definition 3. *Let $f\colon \{0,1\}^n \to \mathbb{R}$. Denote by*

$$\mathrm{OPT}(f) := \max\left\{f(x') \mid x' \in \{0,1\}^n\right\}$$

the maximum value of f, and by

$$T_{(1+1)\text{-}EA,f} := \min\left\{t \in \mathbb{N} \mid f(x^{(t)}) = \mathrm{OPT}(f)\right\}$$

and

$$T_{RLS,f} := \min\left\{t \in \mathbb{N} \mid f(x^{(t)}) = \mathrm{OPT}(f)\right\}$$

the optimization times of the (1+1) EA and RLS, respectively, on f.

Clearly, $T_{(1+1)\text{-EA},f}$ and $T_{\mathrm{RLS},f}$ are random variables. We are mostly interested in their mean values, called *expected optimization time*. Starting one of the algorithms A and letting it run for $T_{A,f}$ steps is called a run of the algorithm. Since the optimization time is non-negative, it follows from Markov's inequality that if the expected optimization time is small then the probability for long runs cannot be close to 1. Large lower bounds on the expected optimization time, however, can be misleading: It is still possible that with probability close to 1 a run will be short. Therefore, in the case of lower bounds, we are also interested in lower bounds for the probability that a single run will take long.

We take the usual approach in the analysis of (randomized) algorithms and concentrate on the asymptotic behavior using the well-known Landau symbols O, Ω, Θ, o, and ω, see, e. g., [1]. We remark that these notions are not only well-defined for functions $t\colon \mathbb{N} \to \mathbb{R}_0^+$ but also for functions $t\colon N \to \mathbb{R}_0^+$ where $N \subseteq \mathbb{N}$ is an infinite set.

To construct the fitness function in Section 4, we use the well-known long k-paths. These paths, that can be easily extended to unimodal functions, were first introduced by Horn, Goldberg, and Deb [6], and later formally defined in the general form by Rudolph [10, 9].

Definition 4. *Let $n \geq 1$. For all $k > 1$ that fulfill $(n-1)/k \in \mathbb{N}$, the long k-path of dimension n, denoted by P_k^n, is a sequence of bit strings from $\{0,1\}^n$ defined as follows. The long k-path of dimension 1 is defined by $P_k^1 := (0,1)$. The long k-path of dimension n is defined via the long k-path of dimension $n-k$, P_k^{n-k}, as follows. Let the long k-path of dimension $n-k$ be given by $P_k^{n-k} = (v_1, \dots, v_l)$. Then we define the sequences of bit strings S_0, B_n, and S_1 from $\{0,1\}^n$, where $S_0 := (0^k v_1,\ 0^k v_2,\ \dots, 0^k v_l)$, $S_1 := (1^k v_l,\ 1^k v_{l-1},\ \dots, 1^k v_1)$, and $B_n := (0^{k-1} 1 v_l,\ 0^{k-2} 11 v_l, \dots, 01^{k-1} v_l)$. The points in B_n build a bridge between the points in S_0 and S_1, that differ in the k leading bits. Therefore, the points in B_n are called* bridge points. *The resulting long k-path P_k^n is constructed by appending S_0, B_n, and S_1, so P_k^n is a sequence of $|P_k^n| = |S_0| + |B_n| + |S_1|$ points. We call $|P_k^n|$ the* length *of P_k^n. The i-th point on the path P_k^n is denoted as p_i, p_{i+j} is called the j-th successor of p_i.*

It is known that the (1+1) EA started in the first half of $P^n_{\sqrt{n-1}}$ (assuming that $(n-1)/\sqrt{n-1} \in \mathbb{N}$ holds) needs an expected number of $\Theta\left(n^{3/2} \cdot 2^{\sqrt{n}}\right)$ steps to reach the path's end. Also, with probability $1 - o(1)$, $o\left(n^{3/2} \cdot 2^{\sqrt{n}}\right)$ steps do not suffice to do so [3].

The recursive definition of long k-paths allows us to determine the length of the paths easily.

Lemma 5. *The long k-path of dimension n has length $|P^n_k| = (k+1)2^{(n-1)/k} - k + 1$. All points of the path are different.*

A proof of this lemma can be found in [10].

The most important property of long k-paths are the regular rules, that hold for the Hamming distances between each point and its successors.

Lemma 6. *Let $n, k \in \mathbb{N}$ be such that the long k-path P^n_k is well defined and let $p_1, \dots, p_{|P^n_k|}$ denote its points. Then for all $1 \le i < j \le |P^n_k|$ with $j - i < k$ the following holds:*

a) The Hamming distance of the points p_i and p_j is $H(p_i, p_j) = j - i$.

b) For all $i < \ell \le |P^n_k|, \ell \ne j$ it holds that $H(p_i, p_\ell) \ne j - i$.

Proof. Again the proof is carried out by induction. The statement is obviously true for $n = 1$ and all values of k. Assume that it holds for P^{n-k}_k. We know that P^n_k is constructed by appending S_0, B_n, and S_1, with $|P^n_k| = (k+1)2^{(n-1)/k} - k + 1$, $|S_0| = |S_1| = (k+1)2^{(n-k-1)/k} - k + 1$, and $|B_n| = k - 1$. We distinguish several cases according to where on P^n_k the points p_i and p_j lie.

If $i < |S_0|$ and $j \le |S_0|$, then the statement holds by assumption, since S_0 has the same structure as P^{n-k}_k.

If $i \le |S_0|$ and $|S_0| < j \le |S_0| + |B_n|$, then the Hamming distance of p_i to the last point in S_0 is $|S_0| - i$ which is at least 0 and less than k by the assumption $j - i < k$. By definition of B_n the ℓ-th point in B_n differs from the last point of S_0 in ℓ bits, so there is exactly 1 point in B_n with Hamming distance $j - i$. All points in S_1 have greater Hamming distance, since the first k bits of points in S_0 and S_1 are all different.

If $|S_0| < i < |S_0| + |B_n|$ and $j \le |S_0| + |B_n|$, the statement is obviously true. All points in B_n just differ on the first k bits, the Hamming distance of any point to its ℓ-th successor is ℓ, so p_i has exactly 1 successor in B_n with Hamming distance $j - i$. All points in S_1 have greater Hamming distance.

If $|S_0| < i \le |S_0| + |B_n|$ and $|S_0| + |B_n| < j$ hold, then the situation is essentially the same as for the second case. The parts S_1 and S_0 have the same structure, only the k leading bits differ and the ordering of S_1 is reversed. So the same remarks apply.

Finally, if $|S_0| + |B_n| < i$, the statement holds by assumption, since S_1 has the same structure as P^{n-k}_k. $\square$

We already mentioned that long k-paths can easily be embedded in a unimodal function. For the sake of completeness, we give a formal definition of unimodality.

Definition 7. *For a function $f: \{0,1\}^n \to \mathbb{R}$, some $x \in \{0,1\}^n$ is called* local optimum *if for all $y \in \{0,1\}^n$ with $H(x, y) = 1$ the inequality $f(x) \ge f(y)$ holds. The function f is called* unimodal *if it has exactly one local optimum. The function f is called* weakly unimodal *if all local optima have equal function value.*

3. EXTREME DIFFERENCES IN PERFORMANCE

In this section, we reiterate some known results on performance differences of RLS and the (1+1) EA. Most of the material is not completely new, but it helps to understand the next section. In particular, we show that RLS can have an exponentially larger expected optimization time even on unimodal functions (that have no local optima other than the global one). We then show an example where RLS with probability $1 - 2^{-\Omega(n)}$ finds the optimum in $\Theta(n^2)$ steps, whereas the (1+1) EA with probability $1 - 2^{-\Omega(n)}$ needs at least 2^n steps.

Since the local mutation is restricted to flipping only single bits it can get stuck in local optima. Standard bit mutation, on the other hand, reaches a global optimum with positive probability from anywhere in the search space. Therefore, it does not come as a surprise that this can lead to extremely large performance differences and enormously different probabilities of finding an optimal point at all. We consider the following fitness function f_1 as a concrete example.

Definition 8. *The function $f_1: \{0,1\}^n \to \mathbb{R}$ is defined by*

$$f_1(x) := \begin{cases} 2 + \sum\limits_{i=1}^{n} x_i & \text{if } \sum\limits_{i=1}^{n} x_i \ne n-1 \\ 1 & \text{otherwise} \end{cases}$$

for any $x \in \{0,1\}^n$.

The function f_1 is known as Jump$_2$ [4]. It has 1^n with $f_1(1^n) = n + 2$ as unique global optimum, all $x \in \{0,1\}^n$ containing exactly 2 0-bits are local optima. It is known and easy to see that $\mathrm{E}\left(T_{(1+1)\ \mathrm{EA}, f_1}\right) = O\left(n^2\right)$ holds [4]. RLS, however, fails to optimize f_1 with probability overwhelmingly close to 1.

Theorem 9. *RLS fails to find the global optimum of f_1 with probability $1 - 2^{-(n-1)}$.*

Proof. RLS cannot reach the global optimum 1^n if some $x \in \{0,1\}^n$ with at least 2 0-bits is reached. Thus, the only way to reach the global optimum is either to have it as initial search point (with probability 2^{-n}) or to have an initial search point with exactly 1 0-bit (with probability $n \cdot 2^{-n}$) and flip exactly the global optimum as first new point (with probability $1/n$). This leads to

$$2^{-n} + n \cdot 2^{-n} \cdot \frac{1}{n} = 2^{-(n-1)}$$

for the probability to find the global optimum. $\square$

It comes as no surprise that local optima pose an obstacle for local search. But if we restrict our attention to (weakly) unimodal functions it is still easy to see that standard bit mutation is able to find short cuts by exploiting some structure in the search space that cannot be exploited by means of mutations of single bits. We consider long 2-paths as a well-known example.

Definition 10. *For $n \in \mathbb{N}$ with $(n-1)/2 \in \mathbb{N}$ the fitness function $f_2\colon \{0,1\}^n \to \mathbb{R}$ is defined by*

$$f_2(x) := \begin{cases} n^2 + l & \text{if } x = p_l \in P_2^n \\ n^2 - n \cdot \left(\sum\limits_{i=1}^{3} x[i]\right) - \sum\limits_{i=4}^{n} x[i] & \text{otherwise} \end{cases}$$

for any $x \in \{0,1\}^n$.

The definition of f_2 coincides with the definition of Path$_2$ in [3]. It is easy to see that f_2 is unimodal: For points $x \notin P_2^n$, it suffices to change any 1-bit to a 0-bit in order to increase the function value. For points $x \in P_2^n$ there is either a neighbor on the path with larger function value or the unique global optimum is reached. Due to symmetry reasons, RLS and the (1+1) EA reach as first point on the path P_2^n some point in the first half of P_2^n with probability at least 1/2. This implies $\Omega\left(n \cdot |P_2^n|\right) = \Omega(n \cdot 2^{n/2})$ as lower bound on $\mathrm{E}\left(T_{\mathrm{RLS},f_2}\right)$. On the other hand, it is easy to see that on average this number of steps is sufficient to optimize f_2, too. The (1+1) EA, however, can reach the optimum of f_2 much faster by using two-bit mutations. It is known [9] that $\mathrm{E}\left(T_{(1+1)\ \mathrm{EA},f_2}\right) = O\left(n^3\right)$ holds. So there is an exponential gap between the expected optimization times of RLS and the (1+1) EA on a unimodal function.

The fitness functions f_1 and f_2 may lead to the impression that standard bit mutation is inherently superior to flipping single bits. This, however, is not true. Considering one-bit flips only may save a search heuristic from reaching regions of the search space that are difficult to leave. A fitness function using this idea has been presented and analyzed in [7] to show that changing the mutation probability in standard bit mutation can cause enormous performance differences. We use a similar function to show a large gap between the performance of RLS and the (1+1) EA.

Definition 11. *The function $f_3\colon \{0,1\}^n \to \mathbb{R}$ is defined by*

$$f_3(x) := \begin{cases} n + 2i & \text{if } x = 1^i 0^{n-i} \\ 3n - 1 & \text{if } x = 1^i 0^j 1 0^k 1 0^{n-i-j-k-2} \text{ with} \\ & (1/4)n \le i \le (3/4)n \text{ and } j,k > 0 \\ n - \sum\limits_{i=1}^{n} x[i] & \text{otherwise} \end{cases}$$

for any $x \in \{0,1\}^n$.

For the vast majority of the search space the function f_3 yields as function value the number of 0-bits. It is well known that both RLS and the (1+1) EA can optimize such a function and find 0^n on average in $O\left(n \log n\right)$ steps. The points $1^i 0^{n-i}$ (with $i \in \{0,1,\ldots,n\}$) form a path with increasing function values starting at 0^n and leading to 1^n, the unique global optimum of f_3. It is well known that both RLS and the (1+1) EA find 1^n in this situation in $O\left(n^2\right)$ steps on average. The points $1^i 0^j 1 0^k 1 0^{n-i-j-k-2}$ (with $(1/4)n \le i \le (3/4)n$ and $j > 0$) are local minima with second best function value. Once such a point is reached, only steps to other such local optima and a direct step to the global optimum 1^n is accepted. Since the Hamming distance between any such point and 1^n is $\Omega(n)$, RLS cannot reach the global optimum and the (1+1) EA needs an exponential number of steps on average and with overwhelming probability. Therefore, it does not make much sense to investigate the expected optimization time. Instead we concentrate on the probability that the optimization time is polynomially bounded.

Theorem 12. *For each constant $c > 2$, it holds that $\mathrm{Prob}\left(T_{RLS,f_3} \le c \cdot n^2\right) = 1 - 2^{-\Omega(n)}$.*

Proof. We describe three events A, B, and C and give lower bounds on their probabilities. It will be clear that if all three events happen, RLS reaches the global optimum of f_3 in at most $c \cdot n^2$ steps. The lower bounds on the probabilities yield upper bounds on the probabilities for the complementary events. The sum of these upper bounds is an upper bound on the probability not to reach the optimum, which completes the proof.

We denote the set of points $1^i 0^{n-i}$ (with $i \in \{0,1,\ldots,n\}$) by P (for path) and the set of points $1^i 0^j 1 0^k 1 0^{n-i-j-k-2}$ (with $(1/4)n \le i \le (3/4)n$ and $j,k > 0$) by L (for local minima). Let A denote the event that the initial search point does not belong to L. Clearly, $\mathrm{Prob}\left(A\right) > 1 - n^3 \cdot 2^{-n}$ holds since we have $|L| = O\left(n^3\right)$.

Let B denote the event that some point in P becomes current search point within the first n^2 steps. If no point in L is hit, the random process of RLS on f_3 before hitting P equals the situation described by the coupon collector's problem [8]: the bits correspond to coupons, each bit is flipped with equal probability $1/n$ (corresponding to obtaining a coupon), we need to flip each bit at most once (corresponding to getting each coupon once). Thus, we know that the probability not to hit P within $n^2 = (n/\log n) \cdot n \log n$ steps is bounded above by $n^{-(n/\log n)+1} = 2^{-n+\log n}$. For each $i \in \{0,1,\ldots,n\}$, we call the set of points with exactly i 0-bits the i-th level. For symmetry reasons, on each level each point has equal probability to become current search point of RLS. Clearly, on each level at most 1 point can become current search point of RLS. The levels containing points in L all have size $2^{\Omega(n)}$ and contain $O\left(n^2\right)$ points belonging to L. Thus, we have $\mathrm{Prob}\left(B\right) = 1 - 2^{-\Omega(n)}$.

Once a point in P is current search point of RLS, no point in L can ever be reached. We reach the unique global optimum when the number of 1-bits is increased to n. For each current search point in P, the probability to increase the number of current 1 bits equals $1/n$. Thus, the expected number of steps needed to reach the global optimum is bounded above by n^2. We consider $(c-1)n^2$ steps. Since we have $c > 2$ we have $c - 1 > 1 + \varepsilon$ for some constant $\varepsilon > 0$. Applying Chernoff bounds [8] we obtain that with probability $1 - 2^{-\Omega(n)}$ the global optimum is reached within $(c-1)n^2$ steps yielding $\mathrm{Prob}\left(C\right) = 1 - 2^{-\Omega(n)}$. □

Clearly, RLS optimizes f_3 efficiently: the probability of a failure is exponentially small. The (1+1) EA, however, is likely to be trapped in the set L of local minima.

Theorem 13. *$\mathrm{Prob}\left(T_{(1+1)\ EA,f_3} < 2^n\right) = 2^{-\Omega(n)}$.*

Proof. We again use the sets P and L introduced in the previous proof. We want to show that with high probability, the (1+1) EA will at some point reach a search point in L. As the search points in L have at least $\frac{1}{4}n - 2$ zeroes, once the (1+1) EA reaches such a search point, it has to flip $\Omega(n)$ bits at once to leave L (and reach the optimal search point). The probability for this to happen is exponentially small.

The probability that the first search point lies on the path P is $n \cdot 2^{-n}$, as $|P| = O\left(n\right)$. Furthermore, it is well known that the first search point will with overwhelming probability have at most $\frac{7}{12}n$ bits set to one. So assume that the first search point has at most this many bits set to one and does neither lie on the path P nor in L. As long as no search point on P is reached, the number of ones of the current search point can only decrease, as this will increase the fitness (alternatively, a search point in L could be reached, which concludes the proof). We now argue that the first search point on the path P reached by the (1+1) EA will with overwhelming probability have at most $\frac{8}{12}n = \frac{2}{3}n$ bits set to one. This is because to reach a search point on P with more bits set, at least $\frac{1}{12}n$ specific bits need to be flipped at once, which happens with probability at most $(1/n)^{-n/12}$.

For each search point on P that has at least $\frac{1}{4}n$ and at most $\frac{3}{4}n$ bits set to one, it suffices to flip any two non-neighboring of the last $\frac{1}{4}n$ bits to reach a point in L. Such a two-bit flip will happen with probability

$$\frac{1}{n^2}\left(1-\frac{1}{n}\right)^{n-2}\sum_{i=1}^{n/4}(i-2) = \Omega(1).$$

We now prove that the (1+1) EA needs $\Omega(n^2)$ mutations with overwhelming probability to reach a search point on P with at least $\frac{3}{4}n$ bits set if it starts from a search point on P that has at most $\frac{2}{3}n$ bits set. This immediately yields that the probability that the (1+1) EA will not stray from the path P is $c^{-\Omega(n^2)}$ for a positive constant $c > 1$.

To increase the number of bits of a search point on P by i, the (1+1) EA has to flip i specific bits at once. The probability for this to happen is

$$\frac{1}{n^i}\cdot\left(\frac{n-1}{n}\right)^{n-i} \le \frac{1}{n^i}\cdot\frac{n-1}{n}$$

for all $i < n$. Hence, we can upper bound the success of each mutation by a geometrically distributed random variable Y_j for which $\text{Prob}\,(Y_j = i) = \frac{1}{n^i}\cdot\frac{n-1}{n}$ holds for all $i \in \mathbb{N}$. The expected value of Y_j is $\text{E}\,(Y_j) = \frac{1}{n-1}$. How much the (1+1) EA increases the number of bits of a search point on P in $t \in \mathbb{N}$ steps is then bounded by the random variable $Y := \sum_{j=1}^{t} Y_j$. If we set $t = \frac{1}{24}n(n-1)$, the expected value of Y is given by $\text{E}\,(Y) = \frac{1}{24}n$. We can now apply the Chernoff bound for geometrically distributed random variables introduced in [2] to see that

$$\text{Prob}\left(Y > \frac{3}{2}\text{E}\,(Y)\right) \le 2^{-\Omega(n^2)}$$

holds. Hence, at least $t \in \Omega(n^2)$ mutations are needed to reach a search point on P with at least $\frac{3}{4}n$ bits set, concluding the proof. $\square$

4. INTUITION AND COUNTER-EXAMPLE

The fitness function f_3 shows a simple reason why a local search may outperform a global search: the global search may be lured into regions of the search space that the local search cannot reach and that are difficult to leave. That this region (the points in L in our example) is difficult to leave for the (1+1) EA came at the price that it cannot be left at all by RLS. This shows that RLS and the (1+1) EA might be affected in different ways by the existence of local optima.

One might hope that without local optima, similar problems cannot occur, and the (1+1) EA is similarly efficient as RLS. In this section, we spoil this hope. We present a weakly unimodal function f_4 that is easily optimized by RLS, but that is very hard for the (1+1) EA. This function is not only uni-modular, but also has the property that RLS finds an optimal solution in time $O(n^2)$ for any initial solution.

More precisely, let $T^x_{A,f}$ denote the optimization time of algorithm A on function f when started with $x \in \{0,1\}^n$ as first search point (instead of random initialization). Then we show the following theorem.

THEOREM 14. *There is a fitness function f_4 such that the following holds.*
1) For all $x \in \{0,1\}^n$, $E(T^x_{RLS,f_4}) = O(n^2)$.
2) With probability $1 - 2^{-\Omega(\sqrt{n})}$, $T_{(1+1)EA,f_4} = 2^{\Omega(\sqrt{n})}$.

The function f_4 will be made precise in Definition 15. The key idea is as follows. Consider some weakly unimodal function with a weakly exponential number of rather short paths leading to a global maximum. Regardless of the starting point, RLS will follow one of these paths and reach a global optimum rather quickly. The (1+1) EA, however, may leave a path by flipping more than a single bit. The function f_4 is designed such that this is likely to happen and that the algorithm is then lead to the beginning of another path. Since there is a weakly exponential number of paths, the (1+1) EA needs on average and with probability very close to 1 a weakly exponential number of steps to reach a global maximum. We proceed by making these ideas concrete.

DEFINITION 15. *The function $f_4\colon \{0,1\}^n \to \mathbb{R}$ is defined for any $n \in \mathbb{N}$ with $n \ge 16$. For such an n we define $n_1 := 4\cdot\lfloor n/8\rfloor$, $k := \lfloor\sqrt{n_1 - 1}\rfloor$, $n_2 := k^2+1$, and $n_3 := n-n_1-n_2$. For $x \in \{0,1\}^n$ we write $x = uvw$ with $u \in \{0,1\}^{n_1}$, $v \in \{0,1\}^{n_2}$, and $w \in \{0,1\}^{n_3}$. For $v \in \{0,1\}^{n_2}$ write $v = v^a v^b$ with $v^a \in \{0,1\}^k$ and $v^b \in \{0,1\}^{n_2-k}$. We consider $P_k^{n_2}$, the long k-path of dimension n_2. Let $(0\ldots0) = p_1, p_2, \ldots, p_l$ with $l = |P_k^{n_2}|$ denote its points.*

Using these notions we define $f_4(x) :=$

$$\begin{cases} 2^n \cdot n^2 & \textit{if } u = 1^{n_1} \textit{ or } v = p_l \\ i\cdot n^2 + j & \textit{if } u = 1^j 0^{n_1-j},\ v = p_i,\ j < n_1 \\ i\cdot n^2 + n + n_1 - j_1 & \textit{if } u = 1^{j_1}0^{j_2}10^{j_3}10^{n_1^*}, \\ & \quad n_1/4 \le j_1 + j_2 \le n_1/2,\ j_2 > 0, \\ & \quad j_1 + j_2 + j_3 \ge (3/4)n_1, \\ & \quad n_1^* = n_1 - j_1 - j_2 - j_3 - 2, \\ & \quad v = p_i \\ i\cdot n^2 + 2n & \textit{if } u = 0^j 10^{n_1-j-1}, \\ & \quad n_1/4 \le j \le n_1/2, \\ & \quad v = p_i,\ i \textit{ odd} \\ i\cdot n^2 - 1 & \textit{if } u = 0^j 10^{n_1-j-1}, \\ & \quad n_1/4 \le j \le n_1/2, \\ & \quad v = p_i,\ i \textit{ even} \\ i\cdot n^2 + 2n & \textit{if } u = 0^j 10^{n_1-j-1}, \\ & \quad j > (3/4)n_1 + 1, \\ & \quad v = p_i,\ i \textit{ even} \\ i\cdot n^2 - 1 & \textit{if } u = 0^j 10^{n_1-j-1}, \\ & \quad j > (3/4)n_1 + 1, \\ & \quad v = p_i,\ i \textit{ odd} \\ n^2 - \left|uv^b\right|_1 - n\left|v^a\right|_1 & \textit{otherwise} \end{cases}$$

for any $x = uvw \in \{0,1\}^n$.

We observe that f_4 is actually well defined: In particular, $P_k^{n_2}$ is well defined since $(n_2-1)/k = k^2/k = k \in \mathbb{N}$ clearly holds. It is easy to see that $n_1 = n/2 - O(1)$, $n_2 = n/2 - O(\sqrt{n})$, and $n_3 = O(\sqrt{n})$ hold. When dividing x into the three parts $x = uvw$, we only care about u and v, the two large parts. The small part w only contains the bits that remain due to our requirements to have n_1 be a multiple of 4 and $\sqrt{n_2-1} \in \mathbb{N}$. Gathering these left-overs in w allows us to define the function f_4 for arbitrary values of n (if they are not too small) and not only for those values of n with $n_3 = 0$, so we do not need to worry whether such values of n do at all exist.

Before giving a formal proof, let us sketch how RLS and the (1+1) EA optimize the function f_4 (Definition 15). RLS can optimize this function easily: As long as the function value is given by $n^2 - |uv^b|_1 - n|v^a|_1$, the function is not harder than a linear function. Thus, on average this region of the search space is left after $O(n \log n)$ steps. After this the function value can always be increased by changing a single bit in u as well as by changing a single bit in v. Changing a single bit in v leads from p_i to p_{i+1} on the long k-path and increases the fitness by n^2. We do not care about these steps. In u, after $O(n)$ changes of a single bit each, a point of the form 1^j0^{n-j} is reached. Once this happens the form of strings in u cannot change any more. After another $O(n)$ changes of single bits in u, a global optimum with function value $2^n \cdot n^2$ is reached. It is not difficult to prove that a global optimum is found on average after $O(n^2)$ steps. Note that this holds regardless of the starting point.

For the (1+1) EA, the situation is different. The main difference is in the situation where $u = 1^j0^{n-j}$ (with $j \le n/3$) and $v = p_i$ holds. We observe that flipping exactly two bits in the right $(n_1/4)$ bits of u and no other bits increases the function value. Since such a step occurs with probability $\Omega(1)$, it is likely that this happens before in u some $1^{j'}0^{n-j'}$ with $j' - j = \omega(1)$ is reached. Since such a step increases the function value there are only two possible ways back to some point of the form 1^h0^{n-h} in u. Either in v we advance from p_i to $p_{i'}$ for some $i' > i$. Such a step has probability $O(1/n)$. Or we reduce the number of 1-bits in u. Since this has a probability of $\Omega(1/n)$, it is likely that this occurs before we advance in v too many times. As the long k-path $P_k^{n_2}$ is weakly exponentially long in n, this is likely to happen a weakly exponential number of times before a global optimum is reached. This implies a weakly exponential lower bound on the expected optimization time of the (1+1) EA on f_4. We now prove these ideas to be correct rigorously.

Proof of Theorem 14. Since the bits in w have no influence on the optimization behavior and $n_3 = o(n)$, we can assume that $n_3 = 0$ for convenience .

First observe that both the (1+1) EA and RLS independent of the initial search point leave the boring area $B := \{x \in \{0,1\}^n \mid f_4(x) < n^2\}$ in expected time $O(n \log n)$. In consequence, with probability $1 - 2^{-\Omega(n/\log n)}$, they find an optimum in time $O(n^2)$. Note that restricted to $B \cup \{0\}$, f_4 is an affine linear function with negative coefficients. Such an objective function, like the more commonly investigated case of positive coefficients, is optimized in expected time $O(n \log n)$ (see e.g. [4, 5]). Hence within this time, the optimum $(0, \dots, 0)$ is found if not some other solution outside B is found before. Since $(0, \dots, 0) \notin B$, this proves the claim.

Run-time analysis for RLS: To prove the statement on RLS, we show that $\mathrm{E}\left(T^x_{\mathrm{RLS},f}\right) = O(n^2)$ for all $x \in \{0,1\}^n \setminus B$. If $x = 1^j0^{n_1-j}p_i$ for some $1 \le j < n_1$ and $1 \le i < l$, then $x = 1^{j+1}0^{n_1-(j+1)}p_i$ and $x = 1^j0^{n_1-j}p_{i+1}$ are the only Hamming neighbors of x not having smaller fitness. Each of them is found with probability $1/n$ in a single mutation step. Hence the expected time to reach a search point of type $x = 1^{j+1}0^{n_1-(j+1)}p_{i'}$ for some i' is $O(n)$. Consequently, $\mathrm{E}\left(T^x_{\mathrm{RLS},f}\right) = O((n_1-j)n) = O(n^2)$.

Now let $x = 1^{j_1}0^{j_2}10^{j_3}10^{n_1-j_1-j_2-j_3-2}p_i$ with $j_1 \ge 1$, $n_1/4 \le j_1 + j_2 \le n_1/2$, $j_1 + j_2 + j_3 \ge (3/4)n_1$ and $1 \le i < l$. Similarly as above, the only Hamming neighbors not having smaller fitness are $x = 1^{j_1-1}0^{j_2+1}10^{j_3}10^{n_1-j_1-j_2-j_3-2}p_i$ and $x = 1^{j_1}0^{j_2}10^{j_3}10^{n_1-j_1-j_2-j_3-2}p_{i+1}$. Consequently, after an expected time of $O(j_1n)$, we find a solution $x' = u'v'w'$ having exactly two ones in u', one at some position j with $n_1/4 + 1 \le j \le n_1/2 + 1$, the other at some position greater than $(3/4)n_1 + 1$. Again there are two Hamming neighbors that can be reached from x', the one by changing one of the two ones to zero (which of the two depends on whether i is even or not), the other by changing $v' = p_{i'}$ to $p_{i'+1}$. Hence after another expected $O(n)$ steps, we end up in one of the cases having exactly one non-zero in the first part of the solution (or with an optimal solution).

Each of the positions x having exactly one one in some position j such that $n_1/4 + 1 \le j \le n_1$ has the following properties. There are only one to three Hamming neighbors having at least the same fitness (hence each is reached with probability $1/n$ in a single mutation). These neighbors are of the following types:

(a) They have two ones as discussed in the previous paragraph,

(b) they have the same u segment as x and an v segment one ahead in the long k-path,

(c) they have only zeroes in the u segment.

Every second search point of type (b) has a neighbor of type (c). In consequence, after an expected number of $O(n)$ steps, RLS finds a search point of type (c).

From a type (c) search point, RLS has a $1/n$ chance to proceed to a search point with first segment 10^{n_1-1}, a $1/n$ chance to move on to another type (c) search point and an $O(1)$ chance reach a type (b) search point. Thus, after $O(n^2)$ steps it actually finds a search point with first segment 10^{n_1-1}. From there, as shown above, it takes $O(n^2)$ steps to find a global optimum.

In summary, for all initial search points x, it takes only an expected number of $O(n^2)$ steps to find an optimal search point.

Run-time analysis for the EA: We now show that with probability $1 - 2^{-\Omega(\sqrt{n})}$, the (1+1) EA needs $\Omega(\sqrt{n})$ steps to find an optimal search point. To ease the language in this proof, we shall say that an event that happens with probability $1 - 2^{-\Omega(\sqrt{n})}$ happens *typically*. Note that, to prove the claim, we may assume $\Omega(\sqrt{n})$ that a typical event holds.

The proof consist of three main arguments. We shall first show that the EA typically finds a search point up_i with $i \le \frac{1}{2}(l+k)$ and $|u|_1 \le 0.2n_1$. We shall then argue that from such a search point, typically no solution x with $|u|_1 > 0.25n_1$ is reached. In consequence, the EA has to reach an optimum of type up_l. We show that typically the EA does not leave the long path encoded in the second segment and only does improvements of less than k steps on the path. Hence finding the end of the path would take at least $\frac{1}{2}(l+k)/k$ steps.

Properties of the first solution outside B: Let $x \in \{0,1\}^n$ be the random initial search point. Typically, x is in the boring area $B := \{x \in \{0,1\}^n \mid f_4(x) < n^2\}$ and satisfies $|v^a|_1 \leq (3/4)k$. As shown in the beginning of the proof, typically the EA needs at most $O(n \log n)$ steps to leave the boring area. We first argue that typically, this results in a solution $y = u_y v_y w_y$ with $|v_y^a|_1 = 0$. Note first that from x with $|v^a|_1 \leq (3/4)k$ only solutions y are accepted that satisfy $|v_y^a|_1 \leq |v^a|_1$ or $|v_y^a|_1 = k$. The probability for the latter to happen in a single mutation is at most $(1/n)^{(1/4)k}$. Hence, typically, no such mutation is found. In consequence, typically, B is left to some position up_i with $i \leq (l+k)/2$.

We shall now argue that the first solution y outside B obtained above typically satisfies $|u|_1 \leq 0.2n_1$. We note that $|u|_1$ may increase due to more-bit mutations that flip some bits in v^a to zero and other bits in u to one. However, the total increase can be bounded by $0.1n_1$: Consider a mutation transforming an $x \in B$ into some other solution y. If $y \in B$ and $|v_y^a|_1 = |v^a|_1$, then $|u_y|_1 \leq |u|_1$. If $|v_y^a|_1 < |v^a|_1$ or $y \notin B$, then $|u_y|_1 = |u|_1 + \sum_{j=1}^{n_1}((u_y)_j - (u)_j)$, where the $((u_y)_j - (u)_j)$ are independent $-1, 0, +1$ random variables with expectation at most $1/n$. Hence a Chernoff bound argument shows that typically, $|u_y|_1 \leq |u|_1 + (1/10)\sqrt{n_1}$. Since there are at most $k \leq \sqrt{n_1}$ such mutations, the total increase of $|u|_1$ through such more-bit mutations is at most $0.1n_1$.

We now estimate the decrease of $|u|_1$ through one-bit mutations. For the initial solution x, we typically have $|u|_1 \leq 0.6n_1$. We first claim that typically it takes at least $100n$ rounds to obtain a solution y such that $v_y^a = 0^j 1^{k-j}$ for some $1 \leq j \leq k$. Fix such a j. Then typically v^a deviates from $0^j 1^{k-j}$ in at least $k/4$ positions. The probability that a fixed such position was flipped (regardless of acceptance) in one of $100n$ mutation steps it at most $1 - (1-1/n)^{100n} \leq 1 - \exp(-100)$. Hence the probability that all the at least $k/4$ positions were flipped in these rounds, is at most $(1-\exp(-100))^{k/4} = 2^{-\Omega(\sqrt{n})}$.

We continue by showing that within these $100n$ rounds, at least once a solution y with $|u_y| < (1/10)n_1$ was reached. Assume not. Then in each round with probability at least $(1/10)(n_1/n)(1-(1/n))^{n-1} \leq (1/20e)(1+o(1))$ a one-bit flip changing a one in the first n_1 bits to zero occurs. The expected number of such mutations would be $(5/e)(1+o(1))n$. Since only $n_1 \leq (1/2)n$ such bits are available, we have deviation from the expectation by $\Theta(n)$. Chernoff bounds show that this occurs with probability $\exp(-\Omega(n))$ only. Hence, typically, at some time a solution y with $|u_y| < (1/10)n_1$ is reached. Since at most $0.1n_1$ new ones are produced by more-bit mutations, we have proven that the first solution $u_y p_i$ outside B satisfies $|u_y|_1 \leq 0.2n_1$ and $i \leq (l+k)/2$.

Up and down. For $u \in \{0,1\}^{n_1}$ write u^a to denote the string of its first $\lfloor n_1/4 \rfloor$ bits. For $x \notin B$ and $|u|_1 \neq n_1$, we denote by $i(x)$ the integer $1 \leq i \leq l$ such that $x = up_i$.

We now analyze how a solution $x \notin B$ with $|u^a|_1 \leq 0.20n_1$ and $i(x) < l$ develops. We call such a solution *upward*, if $u = 1^j 0^{n_1-j}$, and *downward*, if $u = 1^{j_1} 0^{j_2} 1 0^{j_3} 1 0^{n_1-j_1-j_2-j_3-2}$, $n_1/4 \leq j_1 + j_2 \leq n_1/2$, $j_1 +_2 + j_3 \leq (3/4)n_1$, and *turning*, if $u = 0^j 1 0^{n_1-j-1}$, $j \geq n_1/4$.

Let $x \notin B$ and y be the outcome of a single mutation. We call such a mutation exceptional, if y is accepted independent of how the bits in u were mutated. This happens, e.g., if $i(y) > i(x)$ or if x is upward and y is downward.

Let first x be an upward solution and y the outcome of a single mutation step. Then y is downward with constant probability p_{ud}.

Now let x be a downward solution and y the outcome of a single mutation applied to x. Then the probability that y is upward and accepted, is $\Theta(n^{-3})$, as both ones in positions $n_1/4+1, \ldots, n_1$ have to flip and $i(x)$ has to increase. The probability that y is non-exceptional, accepted, and has $|u_y|_1 < |u|_1$ is $\Theta(1/n)$. Now let y be the outcome of applying a sequence of mutations to x until an upward or turning position is reached. We shall call such a sequence of mutations a downward run. Note that the probability that within $\Theta(n^2)$ mutations an upward position was reached, is at most $1-(1-\Theta(n^3))^{\Theta(n^2)} = O(n^{-1})$. Since a non-exceptional mutation in expectation reduces $|u^a|_1$ by at least $1/n$, $\Theta(n^2)$ such mutations with probability $1 - \exp(-\Theta(n))$ suffice to reduce $|u^a|_1$ to 0. Calling such a downward run *successful*, we just showed that a downward run is successful with probability at least $1 - O(n^{-1})$.

We use the above to analyze how a solution $x \notin B$ with $|u^a|_1 \leq 0.2n_1$ and $i(x) < (3/4)l$ develops in a *phase* of several mutations. Let y be the outcome after $\Theta(\sqrt{n})$ changes from upward to downward or turning and back to upward (or vice versa).

We first analyze the increase of $|u^a|_1$ due to mutations generating an upward position out of an upward one or exceptional mutations. Since an upward solution has a constant probability p_{ud} to become downward, typically our $\Theta(\sqrt{n})$ upward runs contain at most $\Omega(\sqrt{n})$ mutations (either exceptional ones or mutations generating an upward position out of an upward one). There are $\Theta(\sqrt{n})$ exceptional mutations marking the turn from an upward run to a downward run and vice versa. Similarly, the downward runs typically in total last $\Theta(n^{2.5})$ steps (ignoring steps that keep a turning position unchanged). The probability for such a mutation to be exceptional is $\Theta(n^{-3})$. Hence again, typically, we have at most $\Theta(\sqrt{n})$ exceptional mutations in the downward runs. In summary, we have $\Theta(\sqrt{n})$ mutations that may increase $|u|_1$. Typically, they will not succeed in increasing $|u|_1$ by say $0.01n_1$.

Typically, at least one of the downward runs is successful. The above shows that, typically, each such phase maintains the property $|u^a|_1 \leq 0.20n$. In particular, typically, within $2^{\Theta(\sqrt{n})}$ steps no solution x with $|u|_1 = n_1$ is generated. In other words, the only optima that can be found within that time are those of type up_l.

Assume that the current search point x is up_i with $i \leq l - k$. By construction of the long k-path, all points on the path succeeding p_i except p_{i+1} to p_{i+k-1} have Hamming distance at least k from p_i. Therefore, we can bound the probability that the EA from x proceeds to some search point $x_1' p_{i'}$ with $i' \geq i + k$ by $2^{-\Theta(\sqrt{n})}$. Hence, typically, the EA does not "leave the path" nor does it make a progress of k or more along the path. In consequence, at least $(l-k)/(2k)$ steps are necessary to find an optimal solution. This concludes the proof. □

We also did experiments to show the concrete behavior of RLS and the (1+1) EA on f_4. Figure 1 shows the average optimization time and standard deviation for search points

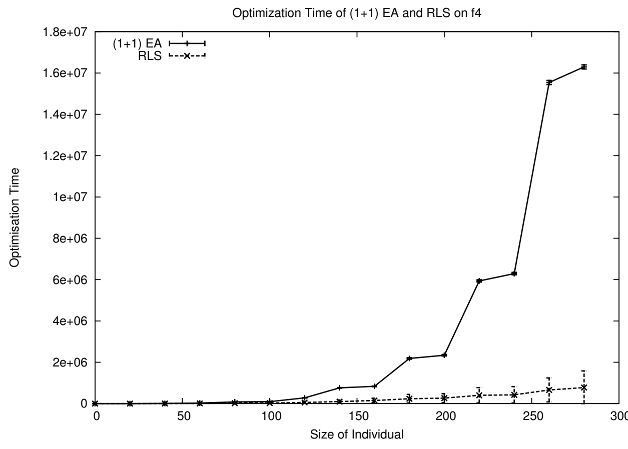

Figure 1: The optimization time needed by the (1+1) EA and by RLS on the function f_4.

of size up to 280 bits averaged over 100 runs. The jumps in the optimization time of the (1+1) EA are caused by the rounded value of the square root in the definition of the value k used by f_4. As soon as the value under the square root gets large enough to reach the next integer, the parameters of the long path increase, causing the increase in runtime.

5. CONCLUSIONS

While the (1+1) EA is one of the simplest evolutionary algorithms possible, randomized local search is an example of a strictly simpler search heuristic that is in many ways similar. Proofs about the performance, however, tend to be very much simpler for randomized local search than for the (1+1) EA due to the global nature of its mutation operator. We investigated the compelling idea of transferring results from RLS to the (1+1) EA and thereby saving a tremendous amount of effort in proofs. Our results show that it is at least very difficult to make such a transfer in non-trivial cases. By presenting illustrative and simple fitness functions we proved that the (1+1) EA can be much superior to RLS due to the existence of local optima as well as the existence of short cuts that cannot be exploited by any local search. On the other hand, we presented an example where RLS almost surely outperforms the (1+1) EA due to a trap in the search space that is almost certainly avoided by RLS but almost unavoidable for the EA. We could prove, however, that such traps are by no means the only reason why the (1+1) EA may be defeated by local search. The existence of a huge number of short paths to global optima can lure the EA into exploring too many of those paths and thus taking an extremely long time while local search stays put on one of these paths reaching a global optimum quickly. The analytical result on the complicated fitness function is accompanied by results of empirical runs demonstrating that this happens even for relatively small dimensions of the search space.

It remains an open problem to find a useful characterization of functions where randomized local search does not outperform the (1+1) EA. It cannot be doubted that such a characterization would be very helpful and mark an important step in the understanding of different randomized search heuristics. Our results, however, establish, that such a characterization is at best very difficult to find.

6. REFERENCES

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. *Introduction to Algorithms.* MIT Press, 2. edition, 2001.

[2] B. Doerr, E. Happ, and C. Klein. A tight bound for the (1+1)-EA on the single source shortest path problem. In D. Srinivasan and L. Wang, editors, *Proceedings of the 2007 IEEE Congress on Evolutionary Computation (CEC)*, pages 1890–1895. IEEE Press, 2007.

[3] S. Droste, T. Jansen, and I. Wegener. On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In A. E. Eiben, T. Bäck, M. Schoenauer, and H.-P. Schwefel, editors, *Proceedings of the 5th International Conference on Parallel Problem Solving From Nature (PPSN)*, volume 1498 of *Lecture Notes in Computer Science*, pages 13–22. Springer, 1998.

[4] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science*, 276(1-2):51–81, 2002.

[5] J. He and X. Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3(1):21–35, 2004.

[6] J. Horn, D. E. Goldberg, and K. Deb. Long path problems. In Y. Davidor, H.-P. Schwefel, and R. Männer, editors, *Proceedings of the Third Conference on Parallel Problem Solving from Nature (PPSN III)*, volume 866 of *Lecture Notes in Computer Science*, pages 149–158. Springer, 1994.

[7] T. Jansen and I. Wegener. On the choice of the mutation probability for the (1+1) ea. In M. Schoenauer, K. Deb, G. Rudolph, X. Yao, E. Lutton, J. J. M. Guervós, and H.-P. Schwefel, editors, *Proceedings of the 6th International Conference on Parallel Problem Solving From Nature (PPSN)*, volume 1917 of *Lecture Notes in Computer Science*, pages 89–98. Springer, 2000.

[8] R. Motwani and P. Raghavan. *Randomized Algorithms.* Cambridge University Press, 1995.

[9] G. Rudolph. How mutation and selection solve long path problems in polynomial expected time. *Evolutionary Computation*, 4(2):195–205, 1996.

[10] G. Rudolph. *Convergence Properties of Evolutionary Algorithms.* Verlag Dr. Kovač, Hamburg, 1997.