Evolutionary Algorithms Are Significantly More Robust to Noise When They Ignore It

Denis Antipov[1] , **Benjamin Doerr**[2] ,
[1]LIP6, CNRS, Sorbonne Université, Paris, France
[2]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
denis.antipov@lip6.fr, doerr@lix.polytechnique.fr

Abstract

Randomized search heuristics (RSHs) are known to have a certain robustness to noise. Mathematical analyses trying to quantify rigorously how robust RSHs are to a noisy access to the objective function typically assume that each solution is re-evaluated whenever it is compared to others. This aims at preventing that a single noisy evaluation has a lasting negative effect, but is computationally expensive and requires the user to foresee that noise is present (as in a noise-free setting, one would never re-evaluate solutions). In this work, we conduct the first mathematical runtime analysis of an evolutionary algorithm solving a single-objective noisy problem without re-evaluations. We prove that the $(1 + 1)$ evolutionary algorithm without re-evaluations can optimize the classic LeadingOnes benchmark with up to constant noise rates, in sharp contrast to the version with re-evaluations, where only noise with rates $O(n^{-2} \log n)$ can be tolerated. This result suggests that re-evaluations are much less needed than what was previously thought, and that they actually can be highly detrimental. The insights from our mathematical proofs indicate that this similar results are plausible for other classic benchmarks.

1 Introduction

In many real-world optimization problems, one does not have a perfect access to the problem instance, but, e.g., the objective function is mildly disturbed by *noise*. Such noise can impose considerable difficulties to classic problem-specific algorithms. Randomized search heuristics, in contrast, are known to be able to cope with certain amounts of stochastic disturbances [Bianchi *et al.*, 2009; Jin and Branke, 2005].

The ability to cope with noise has rigorously been studied and quantified via mathematical runtime analyses [Neumann and Witt, 2010; Auger and Doerr, 2011; Jansen, 2013; Zhou *et al.*, 2019; Doerr and Neumann, 2020], that is, proven performance guarantees for certain algorithms in specific situations. For example, these results have shown that the $(1+1)$ evolutionary algorithm (EA) can solve the classic ONEMAX benchmark defined on bit-strings of length n in polynomial time when noise appears with a rate $O(n^{-1} \log n)$, but the runtime becomes super-polynomial for larger noise rates [Droste, 2004; Gießen and Kötzing, 2016; Dang-Nhu *et al.*, 2018]. For the equally popular benchmark LEADINGONES, only noise rate of order $O(n^{-2} \log n)$ admits a polynomial runtime [Qian *et al.*, 2021; Sudholt, 2021]. Evolutionary algorithms working with larger population sizes tend to be more robust to noise than the $(1 + 1)$ EA, which is essentially a randomized hill-climber [Gießen and Kötzing, 2016; Sudholt, 2021; Antipov *et al.*, 2024].

These and almost all other mathematical runtime analyses of randomized search heuristics in the presence of noise assume that an already constructed solution is re-evaluated whenever it is compared to another solution. This seems to be justified by the fear that a noisy objective value, when not corrected via a renewed evaluation, could harm the optimization process for a long time. Interestingly, the only rigorous support for this fear are two analyses of an ant-colony optimizer in the presence of noise that were provided by Doerr et al. [2012] and by Sudholt and Thyssen [2012], one showing that this algorithm essentially cannot solve stochastic shortest paths problems (without re-evaluations) and the other proving a strong robustness to such disturbances when using re-evaluations. The only other runtime analysis not using re-evaluations is the recent work by Dinot et al. [2023]. Since this work regards a multi-objective optimization problem and attributed the robustness to noise to the implicit diversity mechanisms of the multi-objective evolutionary algorithm regarded, it is hard to predict to what extent the findings generalize to the more classic case of single-objective optimization.

Re-evaluating each solution whenever its objective value is used by the RSH has two disadvantages. The obvious one is the increased computational cost. We note here that usually in black-box optimization, the function evaluations are the computationally most expensive part of the optimization process, up to the point that often the number of function evaluations is used as performance measure. A second problem with assuming re-evaluations in noisy optimization is that this requires the algorithm users to decide beforehand whether they expect to be prone to noise or not – clearly, in a noise-free setting, one would not evaluate a solution more than once.

Our contribution: Given the apparent disadvantages of

re-evaluations and the low theoretical support for these, in this work we study how a simple randomized search heuristic optimizes a classic benchmark problem when not assuming that solutions are re-evaluated. We analyze the setting best-understood in the case of re-evaluations, namely how the $(1+1)$ EA optimizes the LEADINGONES benchmark, for which Sudholt [2021] conducted a very precise runtime analysis, showing among others that a polynomial runtime can only be obtained for noise rates of at most $O(n^{-2}\log n)$. To our surprise, we obtain a much higher robustness to noise when not re-evaluating solutions. We prove that with noise rates up to a constant (which depends on the precise noise model, see Theorems 7 and 9 for the details), the $(1+1)$ EA without re-evaluations optimizes the LEADINGONES benchmark in time quadratic in the problem size n, which is the same asymptotic runtime as in the noise-free setting. This result suggests that the previous strong preference for re-evaluations is not as justified as the literature suggests.

A closer inspection of our proofs also gives some insights in why working with possibly noisy objective values is less detrimental than previously thought, and sometimes even preferable. Very roughly speaking, we observe that noisy function values can also be overcome by generating a solution with true objective value at least as good as the previous noisy function value. Under reasonable assumptions (standard operators and standard noise models with not excessive noise rates), the mutation operator of the evolutionary algorithm has a higher variance than the noise, and consequently, it is easier to correct a noisy objective value by generating a sufficiently good solution than obtaining more noisy objective values due to new noise. From these insights, we are generally optimistic that our findings are not specific to the particular algorithm and benchmark studied in this work.

2 Preliminaries

In this section we define the setting we consider, the notation and also mathematical tools we use in our analysis. In this paper for any pair of integer numbers a, b ($b \geq a$) by $[a..b]$ we denote an integer interval, that is, a set of all integer numbers which are at least a and at most b. If $b < a$, then this denotes an empty set. By $\mathbb{N}$ we denote the set of all strictly positive integer numbers.

2.1 LEADINGONES and Prior Noise

LEADINGONES (LO for brevity) is a benchmark function first proposed by Rudolph [1997], which is defined on bit strings of length n (we call n the *problem size*) and which returns the size of the largest prefix of its argument consisting only of one-bits. More formally, for any bit string x we have

$$\text{LEADINGONES}(x) = \text{LO}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

In this paper we consider optimization of LEADINGONES under prior noise. This means that each time we evaluate the LO value of some bit string x, this bit string is first affected by some stochastic operator N. We call this operator *noise*. Hence, instead of receiving the true value $\text{LO}(x)$ we get $\text{LO}(N(x))$. We consider the following two noise models.

Algorithm 1: The $(1+1)$ EA maximizing a function $f : \{0,1\}^n \to \mathbb{R}$ under noise defined by operator N.

```
// Initialization
Sample x ∈ {0,1}^n uniformly at random;
f_x ← f(N(x));
// Optimization
while not stopped do
    y ← mutate(x);
    f_y ← f(N(y));
    if f_y ≥ f_x then
        x ← y;
        f_x ← f_y;
    end
end
```

One-bit noise. In this noise model with probability q (which is called the *noise rate*) operator $N(x)$ returns a bit string which is different from x in exactly one bit, which is chosen uniformly at random. With probability $1-q$ operator $N(x)$ returns an exact copy of x.

Bitwise noise. In this noise model with *noise rate* $\frac{q}{n}$, operator $N(x)$ flips each bit in x with probability $\frac{q}{n}$ independently from other bits, and returns the resulting bit string. We note that the definition of noise rate is different from the one-bit noise, however in both models the expected number of bits flipped by the noise is equal to q. Also, bitwise noise occurs with probability $1-(1-\frac{q}{n})^n = \Theta(\min(1,q))$, see, e.g., eq. (1) in [Sudholt, 2021].

2.2 The $(1+1)$ EA

We consider a simple elitist evolutionary algorithm called the $(1+1)$ EA. This algorithm stores one individual x (we call it the *parent* individual), which is initialized with a random bit string. Then until some stopping criterion is met[1] it performs iterations, and in each iteration it creates offspring y by applying a mutation operator to x. If the value of the optimized function on y is not worse than its value on x, then y replaces x as the parent for the next iteration. Otherwise x stays as the parent individual. The optimized function is called the *fitness function* and its value on any individual x is called the *fitness* of x. In the rest of the paper we assume that function f optimized by the $(1+1)$ EA is LEADINGONES.

We consider two mutation operators, which in some sense similar to the two noise models. **One-bit mutation** flips exactly one bit chosen uniformly at random. **Standard bit mutation** flips each bit independently from other bits with probability $\frac{\chi}{n}$, where χ is a parameter of the mutation. We call $\frac{\chi}{n}$ the *mutation rate*.

Previous theoretical analyses of the $(1+1)$ EA in noisy environments assumed that the fitness of the parent is re-evaluated in each iteration when it is compared with its offspring. In particular, Sudholt [2021] showed that the $(1+1)$ EA optimizes LEADINGONES in expected time

[1] Similar to many other theoretical studies, we do not define the stopping criterion, but we assume that it does not stop before it finds an optimal solution.

$\Theta(n^2) \cdot e^{\Theta(\min(n, pn^2))}$, where $p < \frac{1}{2}$ is the probability that prior noise occurs. Since for one-bit noise we have $p = q$ and for bitwise noise we have $p = \Theta(\min(1, q))$, this implies that for both noise models with $q = \omega(\frac{\log(n)}{n^2})$, the runtime of the $(1+1)$ EA is super-polynomial.

Since that result by Sudholt [2021] indicated that the $(1+1)$ EA with re-evaluations is not robust to even very small noise rates, we are interested in the behavior of the algorithm when it always uses the first evaluated value for the parent x until this parent is not replaced with a new individual. This approach, in some sense, is similar to using the $(1+1)$ EA on a noisy function without being aware that it is noisy (or just ignoring this fact). The pseudocode of the $(1+1)$ EA using this approach is shown in Algorithm 1.

We enumerate iterations of the $(1+1)$ EA starting from zero, and for all $t \in \mathbb{N} \cup \{0\}$ we use the following notation to describe iteration t. By x_t we denote the parent x at the beginning of iteration t. Slightly abusing the notation, we write $\tilde{f}(x_t)$ to denote the fitness value f_x stored in the algorithm at the beginning of iteration t. By y_t and $\tilde{f}(y_t)$ we denote the offspring y created in iteration t and its noisy fitness f_y correspondingly.

2.3 Auxiliary Tools

In this section we collect mathematical tools which help us in our analysis. We start with the following drift theorem, which is often used in runtime analysis of RSH to estimate the first hitting time of stochastic processes.

Theorem 1 (Additive Drift Theorem by He and Yao [2004], upper bound)**.** *Let $(X_t)_{t\geq 0}$ be a sequence of non-negative random variables with a finite state space $S \subseteq \mathbb{R}_0^+$ such that $0 \in S$. Let $T := \inf\{t \geq 0 \mid X_t = 0\}$. If there exists $\delta > 0$ such that for all $s \in S \setminus \{0\}$ and for all $t \geq 0$ we have $E[X_t - X_{t+1} \mid X_t = s] \geq \delta$, then $E[T] \leq \frac{E[X_0]}{\delta}$.*

We also use the following inequality, which is a simplified version of Wald's equation shown in [Doerr and Künnemann, 2015].

Lemma 2 (Doerr and Künnemann [2015], Lemma 7)**.** *Let T be a random variable with bounded expectation and let $X_1, X_2, \ldots$ be non-negative random variables with $E[X_i \mid T \geq i] \leq C$ for some C and for all $i \in \mathbb{N}$. Then*

$$E\left[\sum_{i=1}^{T} X_i\right] \leq E[T] \cdot C.$$

For any individual x evaluated by the $(1+1)$ EA we call the *active prefix* of x the set of its first $\tilde{f}(x)$ bits, that is, the bits which "pretended" to be ones when we evaluate the fitness of x. The following lemma is an important ingredient of all our proofs.

Lemma 3. *For any individual x and any $i \in [1..\tilde{f}(x)]$ the probability that there are exactly i zero-bits in the active prefix of x is at most the probability that the noise flipped i particular bits when the fitness of x was evaluated. In particular,*

(1) for the one-bit noise with rate q this probability is at most $\frac{q}{n}$ for $i = 1$ and zero for all other i,

(2) for the bitwise noise with rate $\frac{q}{n}$ this probability is at most $(\frac{q}{n})^i$.

Proof. By the definition of the active prefix, when we evaluated the fitness of x, the noise affected it in such way that all its bits in the active prefix became one-bits. Hence if there are i zero-bits in the active prefix of x, all of them have been flipped by the noise before the evaluation. Thus, flipping those i bits is a super-event of the event when we have exactly i zero-bits in the active prefix of x.

For the one-bit noise the probability to flip a particular bit is $\frac{q}{n}$, and it cannot flip more than one bit. For the bitwise noise the probability that it flips i particular bits is $(\frac{q}{n})^i$. □

The next two results are just short mathematical tools, which we formulate as separate lemmas to simplify the arguments in our main proofs. We omit their proofs for reasons of space.[2]

Lemma 4. *For any real values a and b such that $a > b > 0$ and for any positive integer i we have*

$$\frac{a^{i+1} - b^{i+1}}{a^i - b^i} \leq a + b.$$

Lemma 5. *For all $a > 1$ and all integer $n \geq 2$ we have*

$$\sum_{j=2}^{n} \frac{1}{a^j - 1} \leq \frac{\ln\left(1 - \frac{1}{a}\right)}{\ln(a)}.$$

We also use the following inequality which follows from Inequality 3.6.2 by Vasić et al. [2012].

Lemma 6. *For any $n > 0$ and any $x \in [0, n]$ we have*

$$e^x - \frac{x^2}{2n} \leq \left(1 - \frac{x}{n}\right)^n \leq e^x.$$

3 Runtime Analysis

Before starting our theoretical analysis we give an example which provides some intuition behind the effectiveness of the approach with no re-evaluations. Imagine a situation when the current individual x has $\frac{n}{2}$ leading bits, and the algorithm stores its true fitness (that is, $f_x = f(x)$). Assume also that we flip one bit in position $\frac{n}{4}$, and noise flips it back. Then we replace the parent with this offspring, since its noisy fitness is not worse, but we *remember this noisy fitness* and never re-evaluate it later. It means that all future accepted offspring must have their noisy fitness at least as good. This implies that any individual with a bit (or bits) flipped in positions $(n/4..n/2)$ is not accepted, since its fitness is smaller than $n/2$ (unless noise flips the flipped bits and the bit in position $\frac{n}{4}$ back, probability of which is small). This allows us to preserve the accumulated one-bits in those positions until mutation flips the bit in position $n/4$ again and we restore a good individual. On the contrary, the approach with re-evaluation updates the fitness of the parent in one of the next iterations and after that accepts individuals with bits in positions $(n/4..n/2)$ flipped, hence the algorithm loses the accumulated one-bits long before it finds the first improvement by flipping the bit in position $\frac{n}{4}$.

[2] All omitted or sketched proofs can be found in full version of this paper available at arXiv [Antipov and Doerr, 2024].

3.1 One-bit Noise and One-bit Mutation

We start our analysis with the most simple case that we have one-bit noise with rate $q \in (0,1)$ and the $(1+1)$ EA uses one-bit mutation. We aim to estimate the expected number of iterations it takes the algorithm to find the optimum of LEADINGONES when it does not re-evaluate the fitness of the parent solution. The proof idea of this situation will later be used for all other combinations of mutation and noise. The main result of this section is the following theorem.

Theorem 7. *Consider a run of the $(1+1)$ EA with one-bit mutation optimizing* LEADINGONES *under one-bit noise which occurs with probability $q < 1$. Then the EA finds the optimum (the all-ones bit string) and evaluates it properly in expected number of at most $\frac{(1+q)n^2}{(1-q)^2} + \frac{3q}{2(1-q)}$ iterations.*

Before we prove Theorem 7, we need some preparation steps. At the start of any iteration $t \in \mathbb{N}$ the algorithm can be in one of three states: with $f(x_t) = \tilde{f}(x_t)$ (we call this state $S_=$), with $f(x_t) > \tilde{f}(x_t)$ (state $S_>$), or with $f(x_t) < \tilde{f}(x_t)$ (state $S_<$). We divide a run of the $(1+1)$ EA into phases, and each phase (except, probably, the first one) starts in state $S_=$ and ends in the next iteration after which the $(1+1)$ EA starts also in $S_=$.

More formally, phases are defined as follows. Let s_t be the state of the algorithm in iteration t for all $t = 0, 1, \ldots$, and let Q be the set of all iterations, in which the algorithm is in state $S_=$, that is, $Q = \{t \mid s_t = S_=\}$. For all $i \in \mathbb{N}$ let τ_i be the i-th element of Q enumerating them in ascending order. Then for all $i \in \mathbb{N}$ *phase* i is defined as the integer interval $[\tau_i..\tau_{i+1}-1]$. Phase 0 is defined as $[0..\tau_1-1]$. For all i by the *length of phase* i we denote its cardinality, which is $\tau_{i+1} - \tau_i$ for phases $i > 1$ and which is τ_1 for $i = 0$.

An important property of the phases is the *monotonicity* of the true fitness in the beginning of each phase: for each $i \geq 1$ we have $f(x_{\tau_{i+1}}) \geq f(x_{\tau_i})$. This follows from that the noisy fitness f_x stored by the algorithm is never decreased (by condition in line 6 in Algorithm 1 it can be only replaced by a similar or larger value) and from the equality of the noisy and true fitness in the beginning of each phase, that is,

$$f(x_{\tau_{i+1}}) = \tilde{f}(x_{\tau_{i+1}}) \geq \tilde{f}(x_{\tau_i}) = f(x_{\tau_i}).$$

We note that the crucial difference with the approach without re-evaluations is that there this property does not hold, and the example in the beginning of Section 3 demonstrates it.

The following lemma estimates the expected length of one phase.

Lemma 8. *Consider a run of the $(1+1)$ EA with one-bit mutation on* LEADINGONES *under one-bit noise with rate $q < 1$. For all $i \geq 1$ the expected length of phase i is $E[\tau_{i+1} - \tau_i] \leq \frac{1+q}{1-q}$. The expected length of phase 0 is $E[\tau_1] \leq \frac{3q}{2(1-q)}$.*

We only sketch the proof for reasons of space. We first show that the transition probability from state $S_=$ to any other state is at most $\frac{q}{n}$, while the probability to go from states $S_>$ and $S_<$ to state $S_=$ is at least $\frac{1-q}{n}$. Then after the first iteration of the phase it is not finished with probability at most $\frac{2q}{n}$, and it takes at most $\frac{n}{1-q}$ iterations to finish the phase. Hence, the expected phase length is at most $1 + \frac{2q}{1-q} = \frac{1+q}{1-q}$. To obtain the estimate of the expected length of phase 0, we use the law of total expectation over all possible initial states of the algorithm.

With the estimate of the expected time of one phase, we can prove Theorem 7.

Proof of Theorem 7. We define a *super-phase* of the algorithm as follows. Let $R = \{\tau_{i+1} \mid f(x_{\tau_{i+1}}) > f(x_{\tau_i}), i \in \mathbb{N}\}$ that is, R is a set of iterations which start a new phase such that the new phase starts with a strictly higher fitness than the previous phase (in terms of both true and noisy fitness, since they are equal in the beginning of any phase, except phase 0). Note that R has at most n elements, since there are n different fitness values. Let $t_0 = \tau_1$ and for all $i \in [1..|R|]$ let t_i be the i-th element of R, if we sort them in ascending order. Then we define the i-th super-phase as interval $[t_i..t_{i+1}]$ for all $i \in [0..|R|-1]$.

Consider some particular, but arbitrary super-phase i. It consists of one or more phases, and we denote the length of j-th phase in this super-phase by T_j. We call a phase *successful*, if the next phase starts with a strictly better fitness than this phase. This implies that a super-phase ends after a successful phase occurs. A phase is successful, if it consists of one iteration in which mutation flips the first zero-bit of x and noise does not occur. The probability of this event is $\frac{1-q}{n}$. Therefore, the number of phases N in each super-phase is dominated by a geometric distribution $\mathrm{Geom}(\frac{1-q}{n})$. By Lemma 2 and by the estimate of the expected length of a phase from Lemma 8, we have that the expected length of any super-phase is

$$E[t_{i+1} - t_i] = \sum_{j=1}^{N} E[T_j] \leq E[N] \cdot \frac{1+q}{1-q} \leq \frac{(1+q)n}{(1-q)^2}.$$

The total runtime consists of the length of phase 0 and the sum of length of all super-phases. Recalling that the number of super-phases $|R|$ is at most n, by Lemma 2 we obtain that the total runtime is

$$E[T] = E[\tau_1] + \sum_{k=1}^{|R|-1} E[t_{i+1} - t_i]$$
$$\leq \frac{3q}{2(1-q)} + n \cdot \frac{(1+q)n}{(1-q)^2} = \frac{(1+q)n^2}{(1-q)^2} + \frac{3q}{2(1-q)}. \square$$

3.2 Bitwise Noise and Bitwise Mutation

In this section we study the case when the $(1+1)$ EA uses standard mutation and noise is bitwise. This implies that noise has a non-zero probability to flip any number of bits, and thus there might be more than one zero-bit in the active prefix of the current individual x. However, as we show in this section, if noise is not too strong, then the $(1+1)$ EA can handle situations with $k \geq 1$ zero-bits in the active prefix in time of order $O(\frac{1}{p})$, where p is the probability that such situation occurs. This implies that the unfortunate events when the parent has too many zero-bits in its active prefix only add

at most a constant factor to the runtime compared to the noiseless setting. The main result of this section is the following theorem.

Theorem 9. *Consider a run of the $(1+1)$ EA with standard bit mutation with rate $\frac{\chi}{n}$ optimizing* LEADINGONES *under bitwise noise with rate $\frac{q}{n}$. Let $r := \chi + q - \frac{2\chi q}{n}$ and assume that (i) $\chi = \Theta(1)$ and $q = O(1)$ and (ii) there exists a constant $c \in (0,1)$ such that $-\frac{\ln(1-\frac{q}{r})}{\ln \frac{r}{q}} \leq (1-c)e^{-(\chi+q)}$. Then the expected number of iterations until the $(1+1)$ EA finds the optimum (the all-ones bit string) and evaluates it properly is at most*

$$\frac{n^2}{c}\left(\frac{e^{\chi+q}}{\chi} + q\left(\frac{e^{\chi+q}}{\chi}\right)^2\right) + O(n).$$

Before we start the proof, we discuss condition (ii) on χ and q (and r, which is a function of χ and q). If $\chi = \Theta(1)$ and $q = o(1)$, then the left part $-\frac{\ln(1-\frac{q}{r})}{\ln \frac{r}{q}}$ is $o(1)$, while $e^{-(\chi+q)} = \Omega(1)$. Hence, in this case we can choose c close to one, namely $c = 1 - o(1)$. The upper bound on the runtime is then $e^{\chi}\chi^{-1}n^2(1+o(1))$. For larger $q = \Theta(1)$ this condition can be satisfied only for some range of χ. If we express q as a fraction of χ, that is, $q = \alpha\chi$ for some $\alpha > 0$, then condition (ii) can be rewritten as

$$\frac{\ln(1+\alpha)e^{(\alpha+1)\chi}}{\ln(1+\alpha) - \ln\alpha} \leq 1 - c - o(1).$$

From this inequality it trivially follows that to have the left part less than one, we need $\ln\alpha$ to be negative, that is, we need the noise rate to be smaller than the mutation rate. This relation between those two rates ensures that if we get a parent x with wrongly evaluated fitness, then variation of the mutation operator must be stronger than variation of noise, so that fixing this faulty situation was more likely than making it worse. More precise computation[3] allows to see that $\chi \approx 1.4$ allows the left part to be less than one for the maximum possible $q \approx 0.39$. We note, however, that this is likely not a tight bound on the maximum noise rate which can be tolerated by the $(1+1)$ EA.

To prove Theorem 9, we first need several auxiliary results. The next lemma will help to estimate the probability of a "good event", when the algorithm reduces the number of zero-bits in the active prefix of x.

Lemma 10. *Assume that both q and χ are $O(1)$ and let $r = \chi + q - \frac{2q\chi}{n}$ (as in Theorem 9). Let x be some arbitrary bit string. Let $\tilde{y}$ be an offspring of x that was obtained by standard bit mutation with rate $\frac{\chi}{n}$ to x and then bitwise noise with rate $\frac{q}{n}$. Let also S be an arbitrary non-empty subset of $[1..n]$ and let i be its size $|S|$. Consider the event that these three conditions are satisfied:*

(1) each bit with position in S was flipped by exactly one of mutation or noise;

(2) at least one bit with position in S was flipped by mutation; and

(3) all bits with positions not in S have not been flipped.

The probability of this event is at least

$$\left(\frac{r^i - q^i}{n^i}\right)\left(e^{-(\chi+q)} - O\left(\frac{1}{n}\right)\right).$$

We omit its proof for reasons of space.

Similar to the previous section, we distinguish different states in which the algorithm can occur. However, since bitwise noise can lead to any number of zero-bits in the active prefix, we need a more detailed description of state $S_<$ (when the true fitness $f(x_t)$ of the parent is smaller than the stored noisy fitness $\tilde{f}(x_t)$). We define states $S_=$ and $S_>$ similar to the case of one-bit mutation and one-bit noise. Namely, the algorithm is in state $S_=$ in iteration t, if $f(x_t) = \tilde{f}(x_t)$ and it is in state $S_>$, if $f(x_t) > \tilde{f}(x_t)$. For all $j \in [1..n]$ we also say that the algorithm is in state S_j in iteration t, if $f(x_t) < \tilde{f}(x_t)$ and there are exactly j zero-bits in the active prefix of x_t. We divide the run of the algorithm into phases in the similar way as we did in the previous section. Namely, for all $i \in \mathbb{N}$ we define τ_i as the i-th iteration in which the algorithm is in state $S_=$ and we define phase i as an integer interval $[\tau_i..\tau_{i+1}-1]$. Phase 0 is defined as $[0..\tau_1 - 1]$. The length of a phase is its cardinality. Note that the property of the *monotonicity* of the noisy fitness also holds in this case. The following lemma estimates an expected length of a phase, similar to Lemma 8.

Lemma 11. *Consider a run of the $(1+1)$ EA with standard bit mutation with rate $\frac{\chi}{n}$ on* LEADINGONES *under bitwise noise with rate $\frac{q}{n}$. Let $\chi = \Theta(1)$ and $q = O(1)$. Let $r = \chi + q - \frac{2q\chi}{n}$ (that is, $\frac{r}{n}$ is the probability that a particular bit flipped by either mutation or noise, but not by both) and assume that there exists constant $c \in (0,1)$ such that $-\frac{\ln(1-\frac{q}{r})}{\ln \frac{r}{q}} \leq (1-c)e^{-(\chi+q)}$. Then for all $i \in \mathbb{N}$ we have*

$$E[\tau_{i+1} - \tau_i] \leq \frac{1}{c}\left(1 + \frac{qe^{\chi+q}}{\chi}\right) + O\left(\frac{1}{n}\right).$$

The expected length of phase 0 is

$$E[\tau_1] \leq \frac{1}{c}\left((1-c) + \frac{e^{\chi+q}}{2} + \frac{qe^{2(\chi+q)}}{\chi}\right) + O\left(\frac{1}{n}\right).$$

Proof. Since we are aiming at an asymptotic statement, we may assume that n is sufficiently large. To prove this lemma, we use the additive drift theorem. For this reason we assign the following potential Φ to states $S_=$, $S_>$ and all S_j.

$$\Phi(s) = \begin{cases} \frac{n^j}{r^j - q^j}, & \text{if } s = S_j \text{ for all } j \in [1..n], \\ 1 + \frac{qe^{\chi+q}}{r-q}, & \text{if } s = S_>, \\ 0, & \text{if } s = S_=, \end{cases}$$

Consider the process $X_t = \Phi(s_t)$, where s_t is the state of the algorithm in iteration t, and an arbitrary phase i with $i > 0$ which starts at iteration τ_i. Then τ_{i+1} is the first iteration after τ_i where $X_{\tau_{i+1}} = 0$. Note that $X_{\tau_i} = 0$, but in iteration $\tau_i + 1$ we can have larger potentials X_{τ_i+1}. If we find some $\delta > 0$ such that for all $t > \tau_i$ and for all possible values ϕ of the potential we have $E[X_{t+1} - X_t \mid X_t = \phi] \geq \delta$, then by the

[3]The code used to perform these computations can be found in the supplementary material, the file name is "calculate_alpha.py".

additive drift theorem (Theorem 1) we obtain $E[\tau_{i+1} - (\tau_i + 1)] \leq \frac{E[X_{\tau_i+1}]}{\delta}$. Hence, the expected length of the phase is at most $E[\tau_{i+1} - \tau_i] \leq 1 + \frac{E[X_{\tau_i+1}]}{\delta}$.

To estimate $E[X_{t+1} - X_t \mid X_t = \phi]$, we compute the transition probabilities between different states. By Lemma 3, for all $j \in [1..\tilde{f}(x)]$ the probability to go from any state to state S_j is at most $(\frac{q}{n})^j$, and for $j > \tilde{f}(x)$ this probability is zero.

When the algorithm is in state S_j for some $j \in [1..n]$, then to go to a state with a smaller potential (that is, to either $S_=$, $S_>$ or S_k with $k < j$) in one iteration it is sufficient that in y_t the mutation flips at least one zero-bit in the active prefix of x_t (and does not flip any other bit in the active prefix except for other zero-bits) and then, when we evaluate $\tilde{f}(y_t)$, the noise flips all remaining zero-bits in the active prefix (but does not flip any other bit). The probability of this event can be estimated by applying Lemma 10 with S being the set of zero-bits in the active prefix of x_t, that is, this probability is at least

$$\left(\frac{r^j - q^j}{n^j}\right)\left(e^{-(\chi+q)} - O\left(\frac{1}{n}\right)\right).$$

If we go to a state with a smaller potential, then we reduce the potential by at least $\frac{n^j}{r^j-q^j} - \frac{n^{j-1}}{r^{j-1}-q^{j-1}}$, when $j > 1$, and by $\frac{n}{r-q} - 1 - \frac{qe^{\chi+q}}{r-q}$, when $j = 1$. Therefore, when the algorithm is in state S_j with $j \geq 2$, then the drift of the potential is at least

$$\begin{aligned}
&E[X_t - X_{t+1} \mid s_t = S_j, j \geq 2] \\
&\quad \geq \left(\frac{r^j - q^j}{n^j}\right)\left(e^{-(\chi+q)} - O\left(\frac{1}{n}\right)\right) \\
&\qquad \cdot \left(\frac{n^j}{r^j - q^j} - \frac{n^{j-1}}{r^{j-1} - q^{j-1}}\right) - \sum_{k=j+1}^{n} \left(\frac{q}{n}\right)^k \frac{n^k}{r^k - q^k}.
\end{aligned} \tag{1}$$

For reasons of space, we only sketch the estimate of this expression. Using Lemma 4 we can show that the first (positive) term is at least $e^{-(\chi+q)} - O(\frac{1}{n})$, and using Lemma 5 we can show that the absolute value of the second (negative) term is at most $(1-c)e^{-(\chi+q)}$. Hence, we have

$$\begin{aligned}
&E\left[X_t - X_{t+1} \mid s_t = S_j, j \geq 2\right] \\
&\quad \geq e^{-(\chi+q)} - O\left(\frac{1}{n}\right) - (1-c)e^{-(\chi+q)} \\
&\quad = ce^{-(\chi+q)} - O\left(\frac{1}{n}\right).
\end{aligned}$$

For S_1 we can write a similar expression as (1) with a slightly different first positive term, but the same bound will hold. We omit the details for reasons of space. Consequently, the bound

$$E\left[X_t - X_{t+1} \mid s_t = S_j\right] \geq ce^{-(\chi+q)} - O\left(\frac{1}{n}\right)$$

holds for all j.

When the algorithm is in state $S_>$, then to get to state $S_=$ it is enough to flip no bits by mutation or noise. The resulting offspring then is a copy of its parent and its fitness is equal to its noisy fitness, which in $S_>$ is larger than the fitness stored by the algorithm (thus, the new individual replaces the parent). The probability of this event is

$$\left(1 - \frac{\chi}{n}\right)^n \left(1 - \frac{q}{n}\right)^n = e^{-(\chi+q)} - O\left(\frac{1}{n}\right).$$

The transition probabilities from $S_>$ to the states S_i can be computed as before. Hence, the drift in state $S_>$ satisfies

$$\begin{aligned}
&E[X_t - X_{t+1} \mid s_t = S_>] \\
&\quad \geq \left(1 + \frac{qe^{\chi+q}}{r-q}\right)\left(e^{-(\chi+q)} - O\left(\frac{1}{n}\right)\right) \\
&\quad - \sum_{i=1}^{n} \left(\frac{q}{n}\right)^i \frac{n^i}{r^i - q^i} \\
&\quad \geq e^{-(\chi+q)} + \frac{q}{r-q} - O\left(\frac{1}{n}\right) - \frac{q}{r-q} - \sum_{i=2}^{n} \frac{q^i}{r^i - q^i} \\
&\quad \geq e^{-(\chi+q)} - (1-c)e^{-(\chi+q)} - O\left(\frac{1}{n}\right) \\
&\quad = ce^{-(\chi+q)} - O\left(\frac{1}{n}\right),
\end{aligned}$$

where we used Lemma 5 and lemma conditions to obtain the last line.

We have now shown that for every state $s \neq S_=$ the expected progress of X_t is at least $\delta = ce^{-(\chi+q)} - O(\frac{1}{n})$. To apply the additive drift theorem, we now estimate $E[X_{\tau_i+1}]$. For this we note that the probability to go from $S_=$ to any of S_j in one iteration is at most $(\frac{q}{n})^j$ by Lemma 3. To go to state $S_>$ from $S_=$, the algorithm has to create an offspring with true fitness better than the fitness of the current parent. Hence, the probability of this event is at most $\frac{\chi}{n}$ (that is, the probability of flipping the first zero-bit of the parent with mutation). With these estimates of transition probabilities, we obtain

$$\begin{aligned}
E[X_{\tau_i+1}] &\leq \left(1 + \frac{qe^{\chi+q}}{r-q}\right)\frac{\chi}{n} + \sum_{j=1}^{n} \left(\frac{q}{n}\right)^j \frac{n^j}{r^j - q^j} \\
&\leq O\left(\frac{1}{n}\right) + \frac{q}{r-q} + (1-c)e^{-(\chi+q)} = O(1).
\end{aligned}$$

By the additive drift theorem (Theorem 1), we have

$$\begin{aligned}
E[\tau_{i+1} - \tau_i] &\leq 1 + \frac{E[X_{\tau_i+1}]}{\delta} \\
&\leq 1 + \frac{\frac{q}{r-q} + (1-c)e^{-(\chi+q)} + O\left(\frac{1}{n}\right)}{ce^{-(\chi+q)} - O\left(\frac{1}{n}\right)} \\
&= \frac{\frac{q}{r-q} + e^{-(\chi+q)} + O\left(\frac{1}{n}\right)}{ce^{-(\chi+q)} - O\left(\frac{1}{n}\right)} \\
&= \frac{qe^{\chi+q}}{c(r-q)} + \frac{1}{c} + O\left(\frac{1}{n}\right)
\end{aligned}$$

$$= \frac{1}{c}\left(1 + \frac{qe^{\chi+q}}{\chi}\right) + O\left(\frac{1}{n}\right) = O(1).$$

For phase 0 we can use the additive drift theorem with the same potential. We omit this part of the proof for reasons of space. □

We are now in position to prove the main result of this section, Theorem 9.

Sketch of the proof of Theorem 9. For reasons of space and since this proof repeats the proof of Theorem 7, we omit most of the details, except the insignificant differences[4]. We define *super-phases* and *successful phases* in the same way as in the proof of Theorem 7. The probability that a phase is successful is at least the probability that in the first iteration of this phase mutation flips the first zero-bit of an individual and does not flip any other bit, and noise does not flip any bit, which by Lemma 6 is at least $\frac{\chi e^{-(\chi+q)}}{n} - O(\frac{1}{n^2})$. Therefore, the number of phases N in one super-phase is dominated by geometric distribution $\mathrm{Geom}(\frac{\chi e^{-(\chi+q)}}{n} - O(\frac{1}{n^2}))$. Hence, if we denote by T_j the length of j-th phase in a super-phase, then by Lemmas 2 and 11 the expected length of one super-phase is at most

$$E[t_{i+1} - t_i] = \frac{n}{c}\left(\frac{e^{\chi+q}}{\chi} + q\left(\frac{e^{\chi+q}}{\chi}\right)^2\right) + O(1).$$

The optimum is reached after at most n super-phases. If we also take into account phase 0, then the total runtime is then at most

$$E[T] = \frac{n^2}{c}\left(\frac{e^{\chi+q}}{\chi} + q\left(\frac{e^{\chi+q}}{\chi}\right)^2\right) + O(n). \qquad \square$$

3.3 Discussion of the Theoretical Results

When the noise rate is zero, Theorem 7 states that the expected runtime of the algorithm is at most n^2 iterations, and Theorem 9 gives a bound of $\frac{e^{\chi}n^2}{\chi} + O(n)$ iterations (in this case we can chose $c = 1$ to satisfy the assumptions of the theorem). These are larger than the bounds for the $(1+1)$ EA with one-bit mutation and with standard bit mutation shown by Rudolph [1997] and Böttcher et al. [2010], which indicates that our bounds are not tight by at least a constant factor of $\frac{1}{2}$. The main argument in those previous studies which is missing in our analysis is that all bits to the right of the left-most zero-bit in the current parent x are distributed uniformly at random at any time. Therefore, each time the algorithm improves the fitness, the improvement is greater than 1 in expectation (and most of the time it is close to 2). It is not trivial to extend this argument to the noisy case, since situations when the true and noisy fitness of the current individual are not the same might lead to a non-uniform distributions of the bits in the suffix. Nevertheless, we are optimistic that modifying this argument will allow to improve our bounds in future research, and it will also help to prove lower bounds.

It is also interesting to note that when $q = o(1)$, the bounds from Theorems 7 and 9 stay the same as with $q = 0$, apart from a factor of $(1+o(1))$. This suggests that the performance of the $(1+1)$ EA without re-evaluations is not affected by even quite strong noise rates, as long as noise occurs only once in a super-constant number of evaluations on average. However, without a matching lower bound on the noisy runtime we cannot formally state that this suggestion is true.

Finally, we note that the proofs of Theorems 7 and 9 are mostly different in how they treat the noise model, but the choice of the mutation operator is not so important. In all estimates of probabilities of progress (reducing the number of zero-bits in the active prefix of x or having a successful phase) we only want the mutation to flip one particular bit and not to flip any other bits. This implies that the same arguments can be repeated for different combinations of mutation and noise, e.g., for one-bit noise and standard bit mutation or for bitwise noise and one-bit mutation.

4 Conclusion

In this paper we have shown that simple random search heuristics can be very robust to noise, even if a constant fraction of all fitness evaluations are faulty due to the noise. For this, however, it is necessary that the algorithm does not re-evaluate solutions, as if it was unaware of optimizing a noisy function. We showed that this approach leads to a much better performance than the approach dominant in theoretical works, where parent individuals are re-evaluated in each iteration.

Naturally, our mathematical proofs are valid only for the considered settings, namely, for the $(1+1)$ EA with the two mutation operators, the LEADINGONES problem, and the two noise models. For the following reasons we believe that our suggestion to avoid re-evaluations generalizes to broader settings. **First**, wrongfully accepting a bad offspring due to noise is more critical with re-evaluations. When the algorithm re-evaluates such an individual, it starts to re-optimize from this worse point and completely forgets the history of reaching this solution. **Second**, this type of mistake is more likely to happen when we re-evaluate the parent as now a mistake can also be triggered by a noisy evaluation letting the parent appear bad. Without re-evaluations, the only way to accept the wrong solution is that noise lets the offspring appear better than it is. **Third**, when we do not re-evaluate solutions, we keep this bad solution until we fix the mistake with a good mutation. Naturally, if the mutation parameters are chosen well and if the noise is not too strong, undoing a noise-induced error via mutation should be easier that running into such an error. In contrast, with re-evaluations the wrongfully accepted individual can be further mutated to solutions having equal fitness, but being further apart from the original parent. These general working principles are distilled from our analysis, but we believe that they are valid not only in the setting mathematically analyzed in this work.

Acknowledgments

This work was financially supported by the RL4DAC emergence project of Alliance Sorbonne Université. This research also benefited from the support of the FMJH Program PGMO.

[4]The detailed proof of this theorem can also be found in the

References

[Antipov and Doerr, 2024] Denis Antipov and Benjamin Doerr. Evolutionary algorithms are significantly more robust to noise when they ignore it. *CoRR*, abs/2409.00306, 2024.

[Antipov *et al.*, 2024] Denis Antipov, Benjamin Doerr, and Alexandra Ivanova. Already moderate population sizes provably yield strong robustness to noise. In *Genetic and Evolutionary Computation Conference, GECCO 2024*, pages 1524–1532. ACM, 2024.

[Auger and Doerr, 2011] Anne Auger and Benjamin Doerr, editors. *Theory of Randomized Search Heuristics*. World Scientific Publishing, 2011.

[Bianchi *et al.*, 2009] Leonora Bianchi, Marco Dorigo, Luca Maria Gambardella, and Walter J. Gutjahr. A survey on metaheuristics for stochastic combinatorial optimization. *Natural Computing*, 8:239–287, 2009.

[Böttcher *et al.*, 2010] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In *Parallel Problem Solving from Nature, PPSN 2010*, pages 1–10. Springer, 2010.

[Dang-Nhu *et al.*, 2018] Raphaël Dang-Nhu, Thibault Dardinier, Benjamin Doerr, Gautier Izacard, and Dorian Nogneng. A new analysis method for evolutionary optimization of dynamic and noisy objective functions. In *Genetic and Evolutionary Computation Conference, GECCO 2018*, pages 1467–1474. ACM, 2018.

[Dinot *et al.*, 2023] Matthieu Dinot, Benjamin Doerr, Ulysse Hennebelle, and Sebastian Will. Runtime analyses of multi-objective evolutionary algorithms in the presence of noise. In *International Joint Conference on Artificial Intelligence, IJCAI 2023*, pages 5549–5557. ijcai.org, 2023.

[Doerr and Künnemann, 2015] Benjamin Doerr and Marvin Künnemann. Optimizing linear functions with the $(1 + \lambda)$ evolutionary algorithm—different asymptotic runtimes for different instances. *Theoretical Computer Science*, 561:3–23, 2015.

[Doerr and Neumann, 2020] Benjamin Doerr and Frank Neumann, editors. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer, 2020. Also available at http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html.

[Doerr *et al.*, 2012] Benjamin Doerr, Ashish Ranjan Hota, and Timo Kötzing. Ants easily solve stochastic shortest path problems. In *Genetic and Evolutionary Computation Conference, GECCO 2012*, pages 17–24. ACM, 2012.

[Droste, 2004] Stefan Droste. Analysis of the (1+1) EA for a noisy OneMax. In *Genetic and Evolutionary Computation Conference, GECCO 2004*, pages 1088–1099. Springer, 2004.

[Gießen and Kötzing, 2016] Christian Gießen and Timo Kötzing. Robustness of populations in stochastic environments. *Algorithmica*, 75:462–489, 2016.

[He and Yao, 2004] Jun He and Xin Yao. A study of drift analysis for estimating computation time of evolutionary algorithms. *Natural Computing*, 3:21–35, 2004.

[Jansen, 2013] Thomas Jansen. *Analyzing Evolutionary Algorithms – The Computer Science Perspective*. Springer, 2013.

[Jin and Branke, 2005] Yaochu Jin and Jürgen Branke. Evolutionary optimization in uncertain environments – a survey. *IEEE Transactions on Evolutionary Computation*, 9:303–317, 2005.

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

[Qian *et al.*, 2021] Chao Qian, Chao Bian, Yang Yu, Ke Tang, and Xin Yao. Analysis of noisy evolutionary optimization when sampling fails. *Algorithmica*, 83:940–975, 2021.

[Rudolph, 1997] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms*. Verlag Dr. Kovăc, 1997.

[Sudholt and Thyssen, 2012] Dirk Sudholt and Christian Thyssen. A simple ant colony optimizer for stochastic shortest path problems. *Algorithmica*, 64:643–672, 2012.

[Sudholt, 2021] Dirk Sudholt. Analysing the robustness of evolutionary algorithms to noise: refined runtime bounds and an example where noise is beneficial. *Algorithmica*, 83:976–1011, 2021.

[Vasić and Mitrinović, 2012] Petar M. Vasić and Dragoslav S. Mitrinović. *Analytic Inequalities*. Grundlehren der mathematischen Wissenschaften. Springer Berlin Heidelberg, 2012.

[Zhou *et al.*, 2019] Zhi-Hua Zhou, Yang Yu, and Chao Qian. *Evolutionary Learning: Advances in Theories and Algorithms*. Springer, 2019.