Exponential Upper Bounds for the Runtime of Randomized Search Heuristics

Benjamin Doerr(✉)

Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France
doerr@lix.polytechnique.fr

Abstract. We argue that proven exponential upper bounds on runtimes, an established area in classic algorithms, are interesting also in evolutionary computation and we prove several such results. We show that any of the algorithms randomized local search, Metropolis algorithm, simulated annealing, and $(1+1)$ evolutionary algorithm can optimize any pseudo-Boolean weakly monotonic function under a large set of noise assumptions in a runtime that is at most exponential in the problem dimension n. This drastically extends a previous such result, limited to the (1+1) EA, the LeadingOnes function, and one-bit or bit-wise prior noise with noise probability at most 1/2, and at the same time simplifies its proof. With the same general argument, among others, we also derive a sub-exponential upper bound for the runtime of the $(1, \lambda)$ evolutionary algorithm on the OneMax problem when the offspring population size λ is logarithmic, but below the efficiency threshold.

Keywords: Runtime analysis · Noisy optimization · Theory

1 Introduction

The mathematical analysis of runtimes of randomized search heuristics is an established field of the general area of heuristic search [3,15,29,38]. The vast majority of the results in this area show that a certain algorithm can solve (or approximately solve) a certain problem within some polynomial runtime (polynomial upper bound on the runtime) or show that this is not possible by giving a super-polynomial, often exponential, lower bound on the runtime.

As a rare exception to this rule, in his extensive analysis of how the $(1+1)$ evolutionary algorithm ($(1+1)$ EA)[1] optimizes the LeadingOnes benchmark in the presence of prior noise, Sudholt [45, Theorem 6] showed that for one-bit or bit-wise noise with noise probability at most $\frac{1}{2}$, the (1+1) EA finds the optimum of LeadingOnes in time at most $2^{O(n)}$. While clearly a very natural result – everyone would agree that also with such noise the unimodal LeadingOnes

[1] See Section 2 for details on all technical terms used in this introduction.

For reasons of space, some technical details have been omitted from this extended abstract. The interested reader can find them in the extended version [10].

T. Bäck et al. (Eds.): PPSN 2020, LNCS 12270, pp. 619–633, 2020.
https://doi.org/10.1007/978-3-030-58115-2_43

problem should not become harder than the needle-in-the-haystack problem – the technical, long, and problem-specific proof of this result, despite following the intuitive argument just laid out, suggests that such analyses can be harder than one would expect.

In this work, we will argue that such exponential upper bounds are interesting beyond completing a runtime picture of a given problem. We then show that with a different analysis method such uncommon runtime questions can be analyzed relatively easily. As one out of several results, we drastically extend the result in [45] and show that an exponential runtime guarantee holds for

- any of the algorithms randomized local search, Metropolis algorithm, simulated annealing, and $(1+1)$ EA,
- when optimizing any weakly monotonic objective function, e.g., OneMax, linear functions, monotone polynomials, LeadingOnes, plateau functions, and the needle problem,
- in the presence of all common forms of prior and posterior noise with a noise probability of at most $1-\varepsilon$, $\varepsilon > 0$ a constant.

1.1 Exponential Runtime Analysis

The area of mathematical runtime analysis, established as a recognized subfield of the theory of evolutionary algorithms by Ingo Wegener and his research group around twenty years ago, seeks to understand the working principles of evolutionary computation via rigorously proven results on the performance of evolutionary algorithms and other search heuristics in a similar spirit as done in classic algorithms analysis for much longer time.

Adopting the view of classic algorithmics that runtimes polynomial in the problems size are efficient and larger runtimes are inefficient, the vast majority of the results in this field prove polynomial upper bounds or super-polynomial lower bounds. For two reasons, we feel that also super-polynomial and even exponential runtime guarantees are desirable in the theory of evolutionary algorithms.

Our first set of arguments is identical to the arguments made in the classic algorithms field, which led to a shift in paradigms and established the field of exact exponential algorithms [20,21]. These arguments are that (i) for many important problems nothing better than exponential time algorithms are known, so one cannot just ignore these problems in algorithms research, (ii) with the increase of computational power, also exponential time algorithms can be used for problems of moderate (and interesting) size, and (iii) that the existing research on exponential-time algorithms has produced many algorithms that, while still exponential time, are much faster than classic exponential-time approaches like exhaustive search.

Our second line of argument is that exponential time algorithms are of additional interest in evolutionary computation for the following reasons.

(i) *To increase our understanding of the working principles of evolutionary algorithms.* There is a large number of exponential lower bounds in our field, but for essentially none of them an upper bound better than the trivial $n^{O(n)}$

bound exists. It is clear that matching upper and lower bounds tell us most, not only about the runtimes, but also about the working principles of EAs. Tight bounds naturally have to grasp the true way the EA progresses better than loose bounds. For example, the general $n^{O(n)}$ upper bound for all algorithms using standard bit mutation is based on the simple argument that the optimum can be generated from any search point with probability at least n^{-n}. Besides being very pessimistic, this argument does not tell us a lot on how really the EA optimizes the problem at hand (except for the very particular case that the EA is stuck in a local optimum in Hamming distance n to the global optimum). In contrast, as a positive example, the matching $(1 \pm o(1))en \ln n$ upper [35] and lower [24] bound for the runtime of the $(1 + 1)$ EA on OneMax together with their proofs shows that for this optimization process, the effect of mutations flipping more than one bit has no influence on the runtime apart from lower order terms. In a broader sense, this insight suggests that flipping larger number of bits is mainly useful to leave local optima, but not to make fast progress along easy slopes of the fitness landscape.

(ii) *Because understanding runtimes in the exponential and super-exponential regime is important for the application of EAs.* Many classic evolutionary algorithms can easily have a super-exponential runtime. For example, Witt [48] has shown that the simple $(1 + 1)$ EA has an expected runtime[2] of $n^{\Theta(n)}$ on the minimum makespan scheduling problem. Hence knowing that an evolutionary algorithm "only" has an exponential runtime can be interesting.

We note that for problems with exponential-size search spaces (such as the search space $\{0, 1\}^n$ regarded exclusively in this work) blind random search and exhaustive search are exponential-time alternatives. For that reason, in addition to knowing that an EA has an exponential runtime guarantee (that is, a runtime of at most C^n for some constant $C > 1$), it would be very desirable to also have a good estimate for the base of the exponential function, that is, the constant C. Unfortunately, at this moment where we just start reducing the trivial $n^{O(n)}$ upper bound to exponential upper bounds, we are not yet in the position to optimize the constants in the exponent. We are optimistic though (and give some indication for this in Sect. 6) that our methods can be fine-tuned to give interesting values for the base of the exponential function as well. We recall that such an incremental progress is not untypical for the mathematical runtime analysis of EAs – in the regime of polynomial bounds, subject to intensive research since the 1990s, the leading constants for elementary problems such as LeadingOnes and linear functions were only determined from 2010 on [6,44,49].

With this motivation in mind and spurred by the observation that exponential upper bounds are not trivial to obtain, we start in this work a first general attack on the problem of proving exponential upper bounds.

[2] As common both in classic algorithms and in our field, by runtime we mean the worst-case runtime taken over all input instances.

1.2 State of the Art

We are not aware of many previous works on exponential or super-exponential upper bounds on runtimes of EAs. In the maybe first work proving an exponential upper bound, Droste, Jansen, and Wegener [18, Theorem 9] show that the $(1+1)$ EA optimizes the NEEDLE function (called peak function there) in expected time at most $(2\pi)^{-1}n^{1/2}\exp(2n)$. Only a year later, Garnier, Kallel, and Schoenauer [24, Proposition 3.1] in a remarkably precise analysis showed that the expected runtime of the $(1+1)$ EA on the NEEDLE function is $(1 \pm o(1))(1-\frac{1}{e})^{-1}2^n$.

A general upper bound of n^n for the expected runtime of the (1+1) EA on any pseudo-Boolean functions was given in [19, Theorem 6]. Analogous arguments showed an upper bound of $4^n \log_2 n$ for the $(1+1)$ EA using the $2^i/n$ mutation rates in a cyclic fashion [30, Theorem 3] and an upper bound of $O(n^\beta 2^n)$ for the fast $(1+1)$ EA with (constant) power-law exponent $\beta > 1$ [14, Theorem 5.3].

The general $n^{O(n)}$ upper bound of [19] is tight as witnessed, among others, by the trap function [19, Theorem 8] and the minimum makespan scheduling problem [48]. There are a few analyses for parameterized problems showing bounds that can become exponential or worse when the problem parameter is chosen in an extreme manner. Here the $\Theta(n^k)$ runtime bound for the (1+1) EA optimizing jump functions with jump size $k \geq 2$ [19, Theorem 25] is the best known example. More interesting results have been derived in the context of parameterized complexity [37], but again these results have been derived with small parameter values in mind and thus are most interesting for this case.

In contrast to these sporadic upper bounds, there is a large number of exponential lower bounds, e.g., for a broad class of non-elitist algorithms with too low selection pressure [32], for some algorithms using fitness-proportionate selection [26], for the simple genetic algorithm with an only moderately large population size [39], and for various problems in noisy optimization [25,41,45].

Apart from a single exception, for none of these lower bounds it is known whether the runtime is really exponential or is higher, say $n^{\Theta(n)}$. The exceptional exponential upper bound shown in [45, Theorem 6] reads as follows. Consider optimizing the LEADINGONES benchmark function defined on bit strings of length n via the (1+1) EA. Assume that in each iteration, the fitness evaluation of both parent and offspring is subject to stochastically independent prior noise of one of the following two types. (i) With probability $p \leq \frac{1}{2}$, not the true fitness is returned, but the fitness of a random Hamming neighbor. (ii) With probability $p' \in [0,1]$, the search point to be evaluated is disturbed by flipping each bit independently with some probability $q \leq \frac{1}{2}$ and the fitness of this disturbed search point is returned, with probability $1-p'$, the fitness of the original search point is returned; here we assume that $p' \min\{1, qn\} \leq \frac{1}{2}$. Then the expected optimization time, that is, the number of iterations until the optimum is sampled, is at most exponential in n.

With a noise probability of at most $\frac{1}{2}$ and a weakly monotonic, that is, weakly preferring 1-bits over 0-bits, fitness function one would think that this optimization process in some suitable sense is at least as good as the corresponding process

on the NEEDLE function, where absolutely no fitness signal guides the search. This is indeed true, as the proof in [45] shows. Surprisingly, as this proof also shows, it is highly non-trivial to make this intuitive argument mathematically rigorous. The proof in [45] is around four pages long (including the one of the preliminary lemma) and builds on a technical estimate of the mixing time, which heavily exploits characteristics of the LEADINGONES objective function. Consequently, this proof does not easily generalize to other easy benchmark functions such as ONEMAX or linear functions.

1.3 Our Results

Observing that the natural approach taken in [45] is unexpectedly difficult, we develop an alternative approach to proving exponential upper bounds. It builds on the following elementary observation. In the, slightly extremal, situation that we aim at an exponential upper bound, we can wait for an exponentially unlikely "lucky" way to generate the optimum. Being at most exponentially unlikely, that is, having a probability of $p = 2^{-O(n)}$, it takes $2^{O(n)}$ attempts until we succeed. Hence if each attempt takes at most exponential time T_0 (all our attempts will only take polynomial time), we obtain an exponential upper bound on the expected runtime, and moreover, the distributional bound that the runtime is stochastically dominated by T_0 times a geometric distribution with success rate p. This general argument (without the elementary rephrasing in the stochastic domination language) was already used in the proof of the $\text{poly}(n)e^{2n}$ upper bound on the expected runtime of the $(1+1)$ EA on the NEEDLE function by Droste, Jansen, and Wegener [18] more than twenty years ago. It is apparently not very well known in the community, most likely due to the fact that only one year later, Garnier, Kallel, and Schoenauer [24] presented a much tighter analysis of this runtime via different methods. We are not aware of any other use of this argument, which might explain why it was overlooked in [45] (and we give in that we also learned it only very recently).

How powerful this simple approach is, naturally, depends on how easy it is to exhibit lucky ways to find the optimum fast. As we demonstrate, this is in fact often easy. For example (see Theorem 3 for the details), it suffices that in each iteration the probability to move to a Hamming neighbor one step closer to the optimum is $\Omega(n^{-1})$. From this, we can show that from any starting point, the probability to reach the optimum in at most n iterations is at least $2^{-O(n)}$. As argued in the preceding paragraph, this yields an expected runtime of $n2^{O(n)} = 2^{O(n)}$. This argument, without noise and used for the $(1+1)$ EA only, was also used in the NEEDLE analysis in [18].

Together with some elementary computations, this approach suffices to show that a large number of $(1+1)$-type algorithms in the presence of a large variety of types of noise with noise probability at most $1-\varepsilon$, $\varepsilon > 0$ a constant, optimize any weakly monotonic function (including, e.g., ONEMAX, LEADINGONES, and the needle function) in at most exponential time (Theorem 4).

With a few additional arguments, we apply our general approach to a variety of other problems and show exponential upper bounds (i) for the $(1+1)$ EA

optimizing jump functions with jump size at most $\frac{n}{\ln n}$ (Theorem 5), (ii) for any of the above-described algorithms optimizing OneMax in the presence of prior noise flipping each bit independently with probability at most $1-\varepsilon$, where $\varepsilon > 0$ can be any constant (Theorem 6), and (iii) for the $(1+1)$ EA with fitness-proportionate selection optimizing any linear function (Theorem 7). Finally, as an example that our approach can also yield sub-exponential upper bounds, we show that the $(1,\lambda)$ EA with $\lambda \geq (1-\varepsilon)\log_{\frac{e}{e-1}}(n)$, and thus potentially below the threshold for polynomial time, optimizes OneMax in time $\exp(O(n^{\varepsilon}))$ (Theorem 8).

2 Preliminaries

In this section, we briefly describe the algorithms, the noise models, and the benchmark problems considered in this work. We only consider optimization problems defined on the search space $\{0,1\}^n$ of bit strings of length n; we thus also formulate all algorithms only for this setting. We have not doubt, though, that our methods can also be applied to other discrete optimization problems.

We write $[a..b] := \{z \in \mathbb{Z} \mid a \leq z \leq b\}$ and denote by $H(x,y) := |\{i \in [1..n] \mid x_i = y_i\}|$ the *Hamming distance* of two bit strings $x, y \in \{0,1\}^n$. We denote by $\mathrm{Geom}(p)$ the *geometric distribution* with success rate $p \in (0,1]$. Hence if a random variable X is geometrically distributed with parameter p, we write $X \sim \mathrm{Geom}(p)$ to denote this, then $\Pr[X = k] = (1-p)^{k-1}p$ for all $k \in \mathbb{Z}_{\geq 1}$. For two random variables X, Y we write $X \preceq Y$ to denote that Y *stochastically dominates* X, that is, that $\Pr[X \geq \lambda] \leq \Pr[Y \geq \lambda]$ for all $\lambda \in \mathbb{R}$.

Algorithms. We call a randomized search heuristic *single-trajectory* search algorithm if it is an iterative heuristic which starts with a single solution $x^{(0)}$ and in each iteration $t = 1, 2, \ldots$ updates this solution to a solution $x^{(t)}$. We do not make any assumption on how this update is computed. In particular, the next solution may be computed from more than one solution candidate sampled in this iteration. We do, in principle, allow that information other than the search point $x^{(t-1)}$ is taken into iteration t. However, in our main technical result we require that the key condition can be checked only from the search point $x^{(t-1)}$. Formally speaking, this means that for any possibly history of the search process up to this point, when conditioning on this history, the key condition is true. To ease the language, we shall write "regardless of what happened in the first $t-1$ iterations" to express this conditioning.

Examples for single-trajectory algorithms are (randomized) local search, the Metropolis algorithm, simulated annealing, and evolutionary algorithms working with a parent population of size one, such as the $(1+1)$ EA, the fast $(1+1)$ EA [14], $(1+\lambda)$ EA, $(1,\lambda)$ EA, $(1+(\lambda,\lambda))$ GA [11], and SSWM algorithm [40]. We call a single-trajectory algorithm $(1+1)$-*type algorithm* if in each iteration t it generates one solution y and takes as next parent individual $x^{(t)}$ either y or $x^{(t-1)}$. Among the above examples, exactly (randomized) local search, the

Metropolis algorithm, simulated annealing, and the (fast) $(1+1)$ EA are $(1+1)$-type algorithms.

We spare further details on these algorithms and refer the reader to the classic literature for the standard algorithms and to the references given above for the more recent algorithms. For evolutionary algorithms using standard bit mutation, we shall assume that the standard mutation rate $p = \frac{1}{n}$ is used. For our purposes, we mostly need the following property, which in simple words says that the algorithms move to any Hamming neighbor that is not worse than the parent with probability $\Omega(\frac{1}{n})$.

Proposition 1. *For any $(1+1)$-type algorithm A named above (and any choice of the parameters not fixed yet), there is a constant $c_A > 0$ such that the following holds.*

For any iteration t and any z with $H(z, x^{(t-1)}) = 1$, and regardless of what happened in the previous iterations, the offspring y generated by A in iteration t satisfies $\Pr[y = z] \geq \frac{c_A}{n}$. If $f(y) \geq f(x^{(t-1)})$, then also $\Pr[x^{(t)} = z] \geq \frac{c_A}{n}$.

Noise Models. Optimization in the presence of noise, that is, stochastically disturbed access to the problem instance, is an important topic in the optimization of real-world problems. The most common form are noisy objective functions, that is, that the optimization algorithm does not always learn the correct quality (fitness) of a search point. Randomized search heuristics are generally believed to be reasonably robust to noise, see, e.g., [5,31], which differs from problem-specific deterministic algorithms, which often cannot cope with any noise. Some theoretical work exists on how randomized search heuristics cope with noise, started by the seminal paper of Droste [17] and, quite some time later, continued with, among others, [1,4,8,9,16,22,23,25,41,42,45,46]. We refer to the later papers or the survey [36] for a detailed discussion of the state of the art.

In theoretical studies on how randomized search heuristics cope with noise, the usual assumption is that all fitness evaluations are subject to independently sampled noise. Also, it is usually assumed that whenever the fitness of a search point is used, say in a selection step, then it is evaluated anew. In *prior noise* models, the search point x to be evaluated is subject to a stochastic modification and the algorithm learns the fitness f of the disturbed search point (but not the disturbed search point itself). In **one-bit noise with probability** p, with probability p the fitness of a random Hamming neighbor of x is returned, otherwise the correct fitness $f(x)$ is returned. In **independent bit-flip noise with rate** q, from x a search point y is obtained by flipping each bit independently with probability q; then $f(y)$ is returned. In (p, q)**-noise**, with probability p a search point y is obtained from x by flipping each bit independently with probability q and $f(y)$ is returned; otherwise, $f(x)$ is returned.

In the *posterior noise* model, the search point x is first correctly evaluated, but then the obtained fitness $f(x)$ is disturbed. The most common posterior noise is **additive noise**, that is, the returned fitness is $f(x)+X$, where X is a random variable sampled from some given distribution, which does not depend on x (that is, for all search points the difference between the true and the noisy fitness is

identically distributed). The most common setting is that X follows a Gaussian distribution. We note that regardless of X, *independent additive posterior noise gives a correct comparison of two search points of different quality with probability at least* $\frac{1}{2}$.

Since our aim is showing that also in the presence of extreme noise we still have at most exponential runtimes, we also consider the following **unrestricted adversarial noise with probability** p. In this model, with probability $1-p$ the true fitness is returned. With probability p, however, an all-powerful adversary decides the returned fitness value. This adversary knows the algorithm, the optimization problem, and the full history of the optimization process. He does not know, though, the outcome of future random events (both concerning the algorithm and the noise).

Complementing the corresponding statement for posterior noise, the following basic observation estimates the probability that a noisy fitness comparison gives the right result.

Proposition 2. *Let* $\varepsilon > 0$. *Let* $f : \{0,1\}^n \to \mathbb{R}$. *Let* $x, y \in \{0,1\}^n$ *such that* $f(x) \leq f(y)$. *Consider any noise model described above except the one of additive posterior noise. Assume that* $p \leq 1-\varepsilon$ *in the case of one-bit noise or unrestricted adversarial noise,* $(1-q)^n \geq \varepsilon$ *in the case of bit-wise noise,* $1-p(1-(1-q)^n) \geq \varepsilon$ *in the case of* (p,q)*-noise. Denote by* $\tilde{f}$ *the noisy version of* f *with our convention that each noise evaluation of* f *uses fresh independent randomness. Then* $\Pr[\tilde{f}(x) \leq \tilde{f}(y)] \geq \varepsilon^2$.

Proof. Under the conditions named above, with probability at least ε the noisy fitness returns the true fitness value. Consequently, with probability at least ε^2 this happens for both x and y and we have thus $\tilde{f}(x) \leq \tilde{f}(y)$.

Benchmark Problems. We now briefly describe those benchmark problems for which the particular structure is important in the remainder. For further details on these and on all other problems only mentioned in this work, we refer to the literature [3,15,29,38].

As said earlier, we only regard problems defined on bit-strings of length n, hence all functions are $\{0,1\}^n \to \mathbb{R}$. The easiest in many respects benchmark problem is the function **OneMax** defined by $\textsc{OneMax}(x) = \|x\|_1 = \sum_{i=1}^{n} x_i$ for all $x = (x_1, \dots, x_n) \in \{0,1\}^n$. Still unimodal, but not anymore strictly monotonic is the classic **LeadingOnes** function, which counts the number of ones up to the first zero. Formally, $\textsc{LeadingOnes}(x) := \max\{i \in [0..n] \mid \forall j \in [1..i] : x_j = 1\}$. A classic multimodal benchmark is the class of **jump functions**. The jump function with *jump parameter (jump size)* $k \in [1..n]$ is defined by

$$\textsc{Jump}_{nk}(x) = \begin{cases} \|x\|_1 + k & \text{if } \|x\|_1 \in [0..n-k] \cup \{n\}, \\ n - \|x\|_1 & \text{if } \|x\|_1 \in [n-k+1\,..\,n-1]. \end{cases}$$

Hence for $k = 1$, we have a fitness landscape isomorphic to the one of OneMax, but for larger values of k there is a fitness valley ("gap") $G_{nk} := \{x \in \{0,1\}^n \mid n-k < \|x\|_1 < n\}$, which is impossible or hard to cross for most iterative search heuristics.

3 Proving Exponential Upper Bounds

We now state our general technical result which in many situations allows one to prove exponential upper bounds without greater difficulties. We formulate our result for single-trajectory algorithms since this is notationally convenient and covers all our applications (which, in fact, all even concern only $(1+1)$-type algorithms), but we are optimistic that it extends to more general settings. The result is formulated for hitting a general search point x^* as this might turn out to be useful in some applications, but the natural application will be for x^* being the optimum solution.

We remind the reader that the key argument of the proof, running from an arbitrary search point to the target in time $O(n)$ with probability $e^{-O(n)}$, has already appeared in the conference paper [18], but to the best of our knowledge has not been used again since then.

Theorem 3. *Let A be a single-trajectory search algorithm for the optimization of pseudo-Boolean functions. Let $f : \{0,1\}^n \to \mathbb{R}$ and let $x^* \in \{0,1\}^n$. Assume that we use A to optimize f, possible in the presence of noise. Assume that this optimization process satisfies the following property.*

(A) *There is a number $0 < c \leq 1$ such that the following is true. Let $t \geq 1$ and $x, z \in \{0,1\}^n$ such that $x \neq x^*$, $H(x,z) = 1$, and $H(x,x^*) = H(z,x^*) + 1$. Regardless of what happened in the first $t-1$ iterations of optimization process, if $x^{(t-1)} = x$, then $\Pr[x^{(t)} = z] \geq \frac{c}{n}$.*

Let $T = \min\{t \geq 0 \mid x^{(t)} = x^*\}$. Then T is stochastically dominated by $n\,Geom((\frac{c}{e})^n)$. In particular, $E[T] \leq n(\frac{e}{c})^n$.

4 Noisy Optimization of Weakly Monotonic Functions

We now prove that all $(1+1)$-type algorithms discussed in Sect. 2 optimize any weakly monotonic function in at most exponential time even in the presence of any noise discussed in Sect. 2 as long as the noise probability is at most $1-\varepsilon$, $\varepsilon > 0$ a constant, in the cases of prior or adversarial noise. We recall that the only previous result in this direction [45] shows this claim in the particular case of the $(1+1)$ EA optimizing the LeadingOnes function subject to one-bit or (p,q) prior noise with noise probability at most $\frac{1}{2}$.

We say that a function $f : \{0,1\}^n \to \mathbb{R}$ is *weakly monotonic* (or weakly monotonically increasing) if for all $x, y \in \{0,1\}^n$ the condition $x \leq y$ (componentwise) implies $f(x) \leq f(y)$. The class of weakly monotonic functions includes, obviously, all strictly monotonic functions [7,13,28,33] and thus in particular the classic benchmarks OneMax and linear functions with non-negative coefficients [12,19,49]. However, this class also contains more difficult functions like LeadingOnes, monotonic polynomials [47], plateau functions [2], and the needle function.

Theorem 4. *Let $\varepsilon > 0$ be a constant. Let A be one of the randomized search heuristics RLS, the Metropolis algorithm, simulated annealing, or the $(1+1)$ EA using standard bit mutation with mutation rate $\frac{1}{n}$ or using the fast mutation operator with $\beta > 1$. Let $f : \{0,1\}^n \to \mathbb{R}$ be any weakly monotonic function. Assume that A optimizes f under one of the following noise assumptions: one-bit noise or unrestricted adversarial noise with $p \leq 1 - \varepsilon$, bit-wise noise with $(1-q)^n \geq \varepsilon$, (p,q)-noise with $1 - p(1-(1-q)^n) \geq \varepsilon$, or posterior noise with an arbitrary noise distribution.*

Then there is a constant $C > 1$, depending only on ε and the choice of A, such that the time T to sample the optimum $(1,\dots,1)$ of f is stochastically dominated by n Geom(C^{-n}). In particular, the expected optimization time is at most $E[T] \leq nC^n$.

Proof. By Theorem 3, it suffices to show that condition (A) is satisfied for $x^* = (1,\dots,1)$. To this aim, let $x, z \in \{0,1\}^n$ such that $H(x,z) = 1$ and $H(x,x^*) = H(z,x^*) + 1$. Assume that for some iteration t the parent individual satisfies $x^{(t-1)} = x$. By Proposition 1, there is a constant c_A such that the offspring y generated by A in this iteration is equal to z with probability at least $\frac{c_A}{n}$. By the weak monotonicity of f, we have $f(z) \geq f(x)$. By Proposition 2 or the corresponding statement for additive posterior noise, there is a constant $c_N = \min\{\frac{1}{2}, \varepsilon^2\}$ depending on the noise model such that the noisy evaluations of both $x^{(t-1)}$ and $y = z$ in iteration t return an at least as good fitness value for z as for x. In this case, A accepts z with probability one, that is, we have $x^{(t)} = z$. In summary, we have shown $\Pr[x^{(t)} = z] \geq \frac{c_A c_N}{n}$ as desired. Now Theorem 3 immediately gives the claim with $C = \frac{e}{c_A c_N}$.

5 Other Applications of Our Method

To show the versatility of our general approach, we continue with a number of results of varying flavor.

Noisy Optimization of Jump Functions. We first show that the $(1+1)$ EA can optimize noisy jump functions with jump size at most $\frac{n}{\ln n}$ in exponential time. The main argument is that as lucky event we can regard the event that the algorithm progresses towards the optimum by one Hamming step per iteration until the local optimum is reached and then the optimum is reached in one step. The probability of this event is different from the one regarded before and the number of ways to approach the optimum is smaller by a factor of $k!$ (which counts against us), but with the assumption $k \leq \frac{n}{\ln n}$ we obtain the desired exponential runtime.

Theorem 5. *The result of Theorem 4 holds also for the $(1+1)$ EA optimizing* $\textsc{Jump}_{nk}$ *when $k \leq \frac{n}{\ln n}$.*

Optimization of OneMax Under Extreme Bit-Wise Noise. The following result shows that our general method can also exploit particular noise models.

Here, for example, we show that OneMax can be optimized in exponential time even in the presence of bit-wise noise with constant rate $q < 1$. Recall that this means that the search point to be evaluated is disturbed in an expected number of qn bits! To prove this result, we cannot simply invoke Proposition 2, since with probability $1 - o(1)$ the noisy fitness differs from the true fitness. Instead, we show that despite the noise, with probability at least $\frac{1}{2}(1-q)^2$ the better offspring is accepted.

Theorem 6. *Let $\varepsilon > 0$ be a constant. Let A be one of the randomized search heuristics RLS, the Metropolis algorithm, simulated annealing, or the $(1+1)$ EA using standard bit mutation with mutation rate $\frac{1}{n}$ or using the fast mutation operator with $\beta > 1$. Consider optimizing the* OneMax *benchmark function via A in the presence of bit-wise noise with rate $q \leq 1 - \varepsilon$. Then the expected time to find the optimum is at most nK^n, where K is a constant depending on ε and the algorithm used.*

Fitness Proportionate Selection. We now prove an upper bound matching an exponential lower bound proven in [26], namely that the $(1+1)$ EA needs at least exponential time to optimize any linear function with positive coefficients when the usual elitist selection is replaced by fitness-proportionate selection. Here an offspring y of the parent x is accepted with probability $\frac{f(y)}{f(x)+f(y)}$ (and with probability $\frac{1}{2}$ when $f(x)+f(y)=0$). We now show that this result is tight, that is, that an exponential number of iterations suffices to optimize any linear function with this algorithm. This follows easily from Theorem 3 by noting that in the selection step a Hamming neighbor with better fitness is accepted with probability at least $\frac{1}{2}$.

Theorem 7. *Let A be the $(1+1)$ EA with fitness-proportionate selection. Let f be any linear function with positive coefficients. Then the first iteration T in which the optimum of f is generated satisfies $E[T] \leq (2e^2)^n$.*

Subexponential Upper Bounds. Finally, we show that our method is not restricted to showing runtime bounds that are exponential in the problem dimension. We recall that the $(1,\lambda)$ EA is a simple non-elitist algorithm working with a parent population of size one, initialized with a random individual. In each iteration, the algorithm creates independently λ offspring via standard bit mutation (here: with mutation rate $\frac{1}{n}$) and takes a random best offspring as new parent. In their very precise determination of the efficiency threshold of the $(1,\lambda)$ EA on OneMax, Rowe and Sudholt [43] showed that the $(1,\lambda)$ EA has a runtime of at least $\exp(\Omega(n^{\varepsilon/2}))$ when $\lambda \leq (1-\varepsilon)\log_{\frac{e}{e-1}}(n)$, $\varepsilon > 0$ a constant. We now show an upper bound of $\exp(O(n^{\varepsilon}))$ for this runtime. We do not know what is the right asymptotic order of the exponent. From the fact that there is a considerable negative drift when the fitness distance is below $\frac{n^{\varepsilon}}{2\lambda}$, we would rather suspect that also a lower bound of $\exp(\Omega(\frac{n^{\varepsilon}}{\lambda}))$ iterations, and hence $\lambda\exp(\Omega(\frac{n^{\varepsilon}}{\lambda}))$ fitness evaluations, comes true. Since this is not the main topic of this work, we leave this an open problem.

Theorem 8. *Let $0 < \varepsilon < 1$ be a constant. Then there is a constant C_ε such that for all $\lambda \geq (1-\varepsilon)\log_{\frac{e}{e-1}}(n)$ the expected runtime of the $(1,\lambda)$ EA on* OneMax *is at most* $\exp(C_\varepsilon n^\varepsilon)$.

The main proof idea is to first exploit additive drift [27,34] to reach in a short polynomial time of $O(n^{2-\varepsilon})$ a search point in Hamming distance $d_0 = \frac{2e^2 n^\varepsilon}{\lambda}$. We then use an argument analogous to Theorem 3 to show that from such a search point, with probability at least $\exp(-O(n^\varepsilon))$ the optimum is reached in d_0 iterations. This then easily yields the claim.

6 Conclusion and Outlook

In this work, we argued for proving exponential runtime guarantees for evolutionary algorithms. With Theorem 3, we provided a simple and general approach to such problems. It easily gave exponential upper bounds for various algorithmic settings.

In this first work on exponential-time evolutionary algorithms, we have surely not developed the full potential of this perspective in evolutionary computation. The clearly most important question for future work is what can be said about the constant C in the $\text{poly}(n)C^n$ runtime guarantee. A C less than 2 shows that the algorithm is superior to random or exhaustive search. Taking again the field of classic algorithms as example, another interesting question is if there are EAs with "nice" exponential runtimes such as, e.g., the 1.0836^n runtime of the algorithm of Xiao and Nagamochi [50] for finding maximum independent sets in graphs with maximum degree 3.

Concerning the constant C, we note that the proof of [45], which also is not optimized for giving good constants, shows an upper bound that is at least $\exp(3en) \geq (3480)^n$. Under the noise assumptions taken in [45], we have a probability of at least $c_N \geq \frac{1}{4}$ that parent and offspring are not subject to noise. Regarding the $(1+1)$ EA, the probability that a particular Hamming neighbor of the parent is generated as offspring is at least $c_A \geq \frac{1}{en}$. This gives a runtime bound of at most $n(\frac{e}{c_A c_N})^n = n(4e^2)^n \leq n(30)^n$. We are optimistic that with more problem-specific arguments, the constant can be lowered further, possibly below the 2^n barrier. For example, (i) when optimizing any weakly monotonic function subject to 1-bit noise, we accept an offspring strictly dominating the parent (as in the proof of Theorem 3) unless the noise flips a zero-bit of the parent or a one-bit of the offspring. This undesired event happens with probability at most $c_N = \frac{1}{2}$ (instead of $c_N = \frac{1}{4}$), (ii) when optimizing OneMax subject to 1-bit noise, then a better offspring is discarded only if both a one-bit of the offspring and a zero-bit of the parent is flipped. This allows to take $c_N = (1 - O(\frac{1}{n}))\frac{15}{16}$, (iii) when using the $(1+1)$ EA, instead of waiting for the lucky event that in each iteration we approach the target by one Hamming step, we do so with two steps; this reduces the number of different ways to go from a starting point to the optimum by a factor of $2^{n/2}$, but also saves $\frac{n}{2}$ times the factor of $\frac{1}{e}$ for flipping exactly one bit, giving an improvement by a factor of

$(2/e)^{n/2}$. These and further ideas give us some optimism that the constant C can be lowered, possibly to less than 2 (which would prove the algorithm superior to random search).

References

1. Akimoto, Y., Morales, S.A., Teytaud, O.: Analysis of runtime of optimization algorithms for noisy functions over discrete codomains. Theor. Comput. Sci. **605**, 42–50 (2015)
2. Antipov, D., Doerr, B.: Precise runtime analysis for plateaus. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 117–128. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99259-4_10
3. Auger, A., Doerr, B. (eds.): Theory of Randomized Search Heuristics. World Scientific Publishing (2011)
4. Bian, C., Qian, C., Tang, K.: Towards a running time analysis of the (1 + 1)-EA for OneMax and LeadingOnes under general bit-wise noise. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 165–177. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99259-4_14
5. Bianchi, L., Dorigo, M., Gambardella, L.M., Gutjahr, W.J.: A survey on metaheuristics for stochastic combinatorial optimization. Nat. Comput. **8**, 239–287 (2009). https://doi.org/10.1007/s11047-008-9098-4
6. Böttcher, S., Doerr, B., Neumann, F.: Optimal fixed and adaptive mutation rates for the LeadingOnes problem. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 1–10. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_1
7. Colin, S., Doerr, B., Férey, G.: Monotonic functions in EC: anything but monotone! In: Genetic and Evolutionary Computation Conference, GECCO 2014, pp. 753–760. ACM (2014)
8. Dang, D., Lehre, P.K.: Simplified runtime analysis of estimation of distribution algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 513–518. ACM (2015)
9. Dang-Nhu, R., Dardinier, T., Doerr, B., Izacard, G., Nogneng, D.: A new analysis method for evolutionary optimization of dynamic and noisy objective functions. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 1467–1474. ACM (2018)
10. Doerr, B.: Exponential upper bounds for the runtime of randomized search heuristics. CoRR abs/2004.05733 (2020)
11. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theor. Comput. Sci. **567**, 87–104 (2015)
12. Doerr, B., Goldberg, L.A.: Adaptive drift analysis. Algorithmica **65**, 224–250 (2013). https://doi.org/10.1007/s00453-011-9585-3
13. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Mutation rate matters even when optimizing monotone functions. Evol. Comput. **21**, 1–21 (2013)
14. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Genetic and Evolutionary Computation Conference, GECCO 2017, pp. 777–784. ACM (2017)

15. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation-Recent Developments in Discrete Optimization. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4
16. Doerr, B., Sutton, A.M.: When resampling to cope with noise, use median, not mean. In: Genetic and Evolutionary Computation Conference, GECCO 2019, pp. 242–248. ACM (2019)
17. Droste, S.: Analysis of the (1 + 1) EA for a noisy OneMax. In: Deb, K. (ed.) GECCO 2004. LNCS, vol. 3102, pp. 1088–1099. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-24854-5_107
18. Droste, S., Jansen, T., Wegener, I.: On the optimization of unimodal functions with the (1+1) evolutionary algorithm. In: Eiben, A.E., Bäck, T., Schoenauer, M., Schwefel, H.-P. (eds.) PPSN 1998. LNCS, vol. 1498, pp. 13–22. Springer, Heidelberg (1998). https://doi.org/10.1007/BFb0056845
19. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1+1) evolutionary algorithm. Theor. Comput. Sci. **276**, 51–81 (2002)
20. Fomin, F.V., Kaski, P.: Exact exponential algorithms. Commun. ACM **56**, 80–88 (2013)
21. Fomin, F.V., Kratsch, D.: Exact Exponential Algorithms. TTCSAES. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16533-7
22. Friedrich, T., Kötzing, T., Krejca, M.S., Sutton, A.M.: Robustness of ant colony optimization to noise. In: Genetic and Evolutionary Computation Conference, GECCO 2015, pp. 17–24. ACM (2015)
23. Friedrich, T., Kötzing, T., Krejca, M.S., Sutton, A.M.: The compact genetic algorithm is efficient under extreme Gaussian noise. IEEE Trans. Evol. Comput. **21**, 477–490 (2017)
24. Garnier, J., Kallel, L., Schoenauer, M.: Rigorous hitting times for binary mutations. Evol. Comput. **7**, 173–203 (1999)
25. Gießen, C., Kötzing, T.: Robustness of populations in stochastic environments. Algorithmica **75**, 462–489 (2016). https://doi.org/10.1007/s00453-015-0072-0
26. Happ, E., Johannsen, D., Klein, C., Neumann, F.: Rigorous analyses of fitness-proportional selection for optimizing linear functions. In: Genetic and Evolutionary Computation Conference, GECCO 2008, pp. 953–960. ACM (2008)
27. He, J., Yao, X.: Drift analysis and average time complexity of evolutionary algorithms. Artif. Intell. **127**, 51–81 (2001)
28. Jansen, T.: On the brittleness of evolutionary algorithms. In: Stephens, C.R., Toussaint, M., Whitley, D., Stadler, P.F. (eds.) FOGA 2007. LNCS, vol. 4436, pp. 54–69. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-73482-6_4
29. Jansen, T.: Analyzing Evolutionary Algorithms. The Computer Science Perspective. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-17339-4
30. Jansen, T., Wegener, I.: On the analysis of a dynamic evolutionary algorithm. J. Discrete Algorithms **4**, 181–199 (2006)
31. Jin, Y., Branke, J.: Evolutionary optimization in uncertain environments - a survey. IEEE Trans. Evol. Comput. **9**, 303–317 (2005)
32. Lehre, P.K.: Negative drift in populations. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 244–253. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_25
33. Lengler, J.: A general dichotomy of evolutionary algorithms on monotone functions. In: Auger, A., Fonseca, C.M., Lourenço, N., Machado, P., Paquete, L., Whitley, D. (eds.) PPSN 2018. LNCS, vol. 11102, pp. 3–15. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-99259-4_1

34. Lengler, J.: Drift analysis. Theory of Evolutionary Computation. NCS, pp. 89–131. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_2
35. Mühlenbein, H.: How genetic algorithms really work: mutation and hill climbing. In: Parallel problem solving from nature, PPSN 1992, pp. 15–26. Elsevier (1992)
36. Neumann, F., Pourhassan, M., Roostapour, V.: Analysis of evolutionary algorithms in dynamic and stochastic environments. Theory of Evolutionary Computation. NCS, pp. 323–357. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_7
37. Neumann, F., Sutton, A.M.: Parameterized complexity analysis of randomized search heuristics. Theory of Evolutionary Computation. NCS, pp. 213–248. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_4
38. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization -Algorithms and Their Computational Complexity. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-16544-3
39. Oliveto, P.S., Witt, C.: Improved time complexity analysis of the simple genetic algorithm. Theoret. Comput. Sci. **605**, 21–41 (2015)
40. Paixão, T., Heredia, J.P., Sudholt, D., Trubenová, B.: Towards a runtime comparison of natural and artificial evolution. Algorithmica **78**, 681–713 (2017)
41. Qian, C., Bian, C., Jiang, W., Tang, K.: Running time analysis of the $(1+1)$-EA for OneMax and LeadingOnes under bit-wise noise. Algorithmica **81**, 749–795 (2019)
42. Qian, C., Yu, Y., Zhou, Z.: Analyzing evolutionary optimization in noisy environments. Evol. Comput. **26**, 1–41 (2018)
43. Rowe, J.E., Sudholt, D.: The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm. Theoret. Comput. Sci. **545**, 20–38 (2014)
44. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. IEEE Trans. Evol. Comput. **17**, 418–435 (2013)
45. Sudholt, D.: Analysing the robustness of evolutionary algorithms to noise:refined runtime bounds and an example where noise is beneficial. Algorithmica (2020, to appear). https://doi.org/10.1007/s00453-020-00671-0
46. Sudholt, D., Thyssen, C.: A simple ant colony optimizer for stochastic shortest path problems. Algorithmica **64**, 643–672 (2012). https://doi.org/10.1007/s00453-011-9606-2
47. Wegener, I., Witt, C.: On the optimization of monotone polynomials by simple randomized search heuristics. Comb. Probab. Comput. **14**, 225–247 (2005)
48. Witt, C.: Worst-case and average-case approximations by simple randomized search heuristics. In: Diekert, V., Durand, B. (eds.) STACS 2005. LNCS, vol. 3404, pp. 44–56. Springer, Heidelberg (2005). https://doi.org/10.1007/978-3-540-31856-9_4
49. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**, 294–318 (2013)
50. Xiao, M., Nagamochi, H.: Confining sets and avoiding bottleneck cases: a simple maximum independent set algorithm in degree-3 graphs. Theoret. Comput. Sci. **469**, 92–104 (2013)