A General Dichotomy of Evolutionary Algorithms on Monotone Functions

Johannes Lengler

***Abstract*—It is known that the $(1+1)$-EA with mutation rate c/n optimizes every monotone function efficiently if $c < 1$, and needs exponential time on some monotone functions (HOTTOPIC functions) if $c \geq 2.2$. We study the same question for a large variety of algorithms, particularly for the $(1+\lambda)$-EA, $(\mu+1)$-EA, $(\mu+1)$-GA, their "fast" counterparts, and for the $(1+(\lambda,\lambda))$-GA. We find that all considered mutation-based algorithms show a similar dichotomy for HOTTOPIC functions, or even for all monotone functions. For the $(1+(\lambda,\lambda))$-GA, this dichotomy is in the parameter $c\gamma$, which is the expected number of bit flips in an individual after mutation and crossover, neglecting selection. For the fast algorithms, the dichotomy is in m_2/m_1, where m_1 and m_2 are the first and second falling moment of the number of bit flips. Surprisingly, the range of efficient parameters is not affected by either population size μ nor by the offspring population size λ. The picture changes completely if crossover is allowed. The genetic algorithms $(\mu+1)$-GA and $(\mu+1)$-fGA are efficient for arbitrary mutations strengths if μ is large enough.**

***Index Terms*—Computational and artificial intelligence, evolutionary computation, genetic algorithms.**

I. INTRODUCTION

FOR EVOLUTIONARY algorithms (EAs), choosing a good mutation strength is a delicate matter that is subject to conflicting goals. For example, consider a pseudo-Boolean fitness function $f : \{0,1\}^n \to \mathbb{R}$ and an EA with standard bit mutation, i.e., all bits are flipped independently. On the one hand, if the mutation strength is too low then the progress is also slow, and the algorithm will be susceptible to local optima. On the other hand, if the mutation rate is too high and the parent is close to a global optimum then typically the offspring, even if it has a "good" mutation in it, will also have a large number of detrimental mutations. A well-known example of this tradeoff are linear functions (e.g., ONEMAX), for which there is an optimal mutation rate $1/n$: this rate minimizes the expected runtime, i.e., the expected number of function evaluations before the optimum is found, up to minor order terms [1], [32]. Any deviation from this mutation rate by a constant factor to either direction decreases the performance.

A different, more extreme example are (strictly) monotone pseudo-Boolean functions.[1] A function $f : \{0,1\}^n \to \mathbb{R}$ is *strictly monotone* if, for every $x, y \in \{0,1\}^n$ with $x \neq y$ and such that $x_i \geq y_i$ for all $1 \leq i \leq n$, it holds that $f(x) > f(y)$. In particular, every monotone function has a unique global optimum at $(1 \dots 1)$. Moreover, every such function is efficiently optimized by random local search (RLS), which is the $(1+1)$-type algorithm that flips in each round exactly one bit, chosen uniformly at random. From any starting point, RLS finds the optimum with at most n improving steps, and by a coupon collector argument it optimizes any monotone function in $O(n \log n)$ steps. Thus, monotone functions may be considered trivial to optimize, and we might expect every EA to solve them efficiently.

However, this is not so. Doerr *et al.* [9], [10] showed that even the $(1+1)$ EA $(1+1)$-EA, which flips each bit independently with static mutation rate c/n, may have problems. More precisely, for small mutation rate, $c < 1$, the $(1+1)$-EA has expected runtime $O(n \log n)$ as desired,[2] but for large mutation rate, $c > 16$, there are monotone functions for which the $(1+1)$-EA needs exponential time. The construction in [9] and [10] was based on exponentially long paths in $\{0,1\}^n$. Lengler and Steger [22] gave a simpler construction of such "hard" monotone functions, which we call HOTTOPIC (they did not provide a name), and which yield exponential runtime for $c \geq c_0 := 2.13692\dots$ The basic idea of this construction is that there is some subset of bits which form a "hot topic," i.e., the algorithm considers them much more important than the other bits. This hot topic is different for different regions of the search space, and thus the hot topic changes when the algorithms moves into a new region of the search space. An algorithm with a large mutation rate that focuses too much on the current hot topic tends to deteriorate the quality of the remaining bits. If the hot topic changes often, then the algorithm stagnates.

Since both low and high mutation rates have their disadvantages, many different strategies have been developed to gain the best of two worlds. In this paper, we pick a collection of either traditional or particularly promising methods, and analyze whether they can overcome the detrimental effect of the

Manuscript received June 27, 2018; revised October 29, 2018 and February 13, 2019; accepted May 9, 2019. Date of publication May 15, 2019; date of current version December 1, 2020.

The author is with the Department of Computer Science, ETH Zürich, 8092 Zürich, Switzerland (e-mail: johannes.lengler@inf.ethz.ch).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the author.

Digital Object Identifier 10.1109/TEVC.2019.2917014

[1] We will be sloppy and drop the term "strictly," but throughout this paper, we always mean strictly monotone functions.

[2] This result was already implicit in [18], and in a very recent preprint by Lengler *et al.* it was shown that there is $\varepsilon > 0$ such that the runtime is $O(n \log^2 n)$ for all $c \leq 1+\varepsilon$ [21].

HOTTOPIC functions.[3] In particular, we consider (for constant μ, λ, and for static parameters) the classical $(1+\lambda)$-EA, $(\mu+1)$-EA, and $(\mu+1)$-GA, the $(1+(\lambda,\lambda))$-GA by Doerr, Doerr, and Ebel [8], and the recently proposed "fast $(1+\lambda)$-EA," "fast $(\mu+1)$-EA," and "fast $(\mu+1)$-GA" [12], which we denote by $(1+1)$-fEA, $(1+\lambda)$-fEA and $(\mu+1)$-fGA, respectively. Surprisingly, for mutation-based algorithms neither μ nor λ have any effect on the results. While we do obtain a fine-grained landscape of results (see below), one major trend is prevailing: crossover helps!

An extended abstract of this paper has been presented at the 15th International Conference on Parallel Problem Solving from Nature (PPSN'18) [20]. Some parts had to be omitted from the print version, including the full proof of Theorem 5. They can be found it in the online supplementary material.

A. Results

In this section, we collect our results for the different algorithms. An overview can be found in Table I. Note that, unless explicitly otherwise stated, we always assume that the parameters μ, λ, c, γ of the algorithms are constant.

1) Classical EAs: For the classical EA $(1+\lambda)$-EA, we show a dichotomy: if the mutation parameter c is sufficiently small, then the algorithms optimize all monotone functions in time $O(n \log n)$, while for large c the algorithm needs exponential time on some HOTTOPIC functions. The interesting question is: how does the threshold for c depend on the parameters λ? It may seem that a large λ bears some similarity with an increased mutation rate. After all, the total number of mutations in each generation is increased by a factor of λ. Thus, we might expect that the $(1+\lambda)$-EA has difficulties with monotone functions for even smaller values of c. However, this is not so. The bounds on the mutation rate, $c < 1$, and $c > c_0 = 2.13\ldots$ do *not* depend on λ. In fact, for the HOTTOPIC functions we can show that this is tight: if $c < c_0$ then the $(1+\lambda)$-EA and the $(\mu+1)$-GA optimize all HOTTOPIC functions in time $O(n \log n)$, while for $c > c_0$ it is exponentially slow on some HOTTOPIC instances.

For the $(\mu+1)$-EA we get the same result on HOTTOPIC. In particular, the threshold on c is also independent of μ. For the $(\mu+1)$-EA, we could not show an upper runtime bound for *all* monotone functions in the case $c < 1$, so currently we cannot exclude that the situation might get even *worse* for larger μ, as there might still be other monotone functions which are hard for the $(\mu+1)$-EA with $c < 1$.

2) $(\mu+1)$-GA: It has been observed before that some algorithms may be sped up by crossover, e.g., if we switch from the $(\mu+1)$-EA to the $(\mu+1)$-GA. In particular, Sudholt [31] and Corus and Oliveto [2] showed that the $(\mu+1)$-GA is by a constant factor faster than the $(\mu+1)$-EA on ONEMAX. For monotone functions we also observe a change, but in extremis. We show that for the HOTTOPIC functions crossover extends the range of mutation rate arbitrarily. For every $c > 0$, if μ is a sufficiently large constant then the $(\mu+1)$-GA finds the optimum of HOTTOPIC in time $O(n \log n)$. At present, there are no monotone functions known on which the $(\mu+1)$-GA with arbitrary c and large $\mu = \mu(c)$ is slow. It remains an intriguing open question whether the $(\mu+1)$-GA with large μ is fast on *every* monotone function.

3) $(1+(\lambda,\lambda))$-GA: This algorithm creates λ offspring, and uses the best of them to perform λ biased crossovers with the parent, see Section II-B. The best crossover offspring is then compared with the parent. This algorithm has been derived by Doerr *et al.* [7], [8] from a theoretical understanding of so-called *black-box complexity*, and has been intensively studied thereafter [3]–[6]. Most remarkably, it gives an asymptotic improvement on the runtime of the most intensively studied test function ONEMAX, on which it can achieve runtime roughly $n\sqrt{\log n}$ for static settings (up to $\log \log n$ terms), and linear runtime $O(n)$ for dynamic parameter settings. These runtimes are achieved with a nonconstant $\lambda = \lambda(n)$. Apart from some highly artificial constructions, the $(1+(\lambda,\lambda))$-GA is the only known unbiased EA that can optimize ONEMAX faster than $\Theta(n \log n)$.

The algorithm comes with three parameters: the offspring population size λ, the mutation rate c/n by which the offspring are created, and a crossover bias γ, which is the probability to take the offspring's genes in the crossover. Again we find a dichotomy between weak and strong mutation, but this time not in c, but rather in the product $c\gamma$. In [6], it is suggested to choose c, γ such that $c\gamma = 1$. Note that this makes sense, because $c\gamma$ is (up to possible biases by the selection process) the expected number of mutations in the crossover child. Thus, it is plausible that it plays a similar role as the parameter c in classical algorithms. Indeed we find that for $c\gamma < 1$ the runtime is small for every monotone function, while for $c\gamma > c_0 = 2.13\ldots$ it is exponential on HOTTOPIC functions. As before, the bound is tight for HOTTOPIC, i.e., for $c\gamma < c_0$ the $(1+(\lambda,\lambda))$-GA needs time $O(n \log n)$ to optimize HOTTOPIC.

Notably, the runtime benefits on ONEMAX carry over to the HOTTOPIC function. Since the benefits on ONEMAX in previous work have been achieved for nonconstant parameter choices, we relax our assumption on constant parameters for the $(1+(\lambda,\lambda))$-GA. More precisely, we show that if $\varepsilon < c\gamma < 1-\varepsilon$ for a constant $\varepsilon > 0$, then for *any* choice of c, γ, λ (including nonconstant and/or adaptive choices), the $(1+(\lambda,\lambda))$-GA optimizes every monotone function in $O(n \log n)$ generations. Moreover, we show that for the optimal static parameter and adaptive parameter settings in [6], the algorithm achieves the same asymptotic runtime on HOTTOPIC as on ONEMAX, in particular runtime $O(n)$ in the adaptive setup.[4]

Unfortunately, it seems unlikely that the runtimes of $o(n \log n)$ for ONEMAX carry over to arbitrary monotone functions, because they are achieved by increasing c and λ with n (although $c\gamma$ is left constant). For ONEMAX, if there is a zero-bit that is flipped in one of the mutations, then at

[3]Other methods include parameter control, i.e., dynamic or adaptive choice of parameters. We will discuss them briefly in Section I-B. In this paper, we are mostly concerned with static parameters.

[4]Strictly speaking, the adaptive parameter choice is not natural for HOTTOPIC, since the parameters must be chosen as a function of the remaining zero-bits in the search points. For HOTTOPIC, or for general monotone functions, this information is not naturally available. However, in [6] it was shown that the same effect can be achieved for the $(1+(\lambda,\lambda))$-GA by an adaptive (self-adjusting) setup using the one-fifth rule, which is applicable for monotone functions.

TABLE I
OVERVIEW OVER THE RESULTS OF THIS PAPER. THE DEFINITION OF $c_0 = 2.13692\ldots$ CAN BE FOUND IN THEOREM 6. EACH ENTRY GIVES A SUFFICIENT CONDITION FOR THE RUNTIME STATEMENT OF THE CORRESPONDING COLUMN. IF SEVERAL LINES ARE IN ONE CELL, THEN, EACH LINE IS A SUFFICIENT CONDITION. UNLESS OTHERWISE STATED, $c, \lambda, \mu = \Theta(1)$. ALL RESULTS EXCEPT FOR THE $(1+1)$-EA ARE PROVEN IN THIS PAPER. THE RESULTS OF THE FIRST COLUMN ARE IN THEOREMS 3 AND 4, THE OTHER RESULTS ARE IN THEOREM 6, EXCEPT THAT REMARK C IS IN THEOREM 3

Algorithm	$O(n \log n)$ on mon. funct's	$O(n \log n)$ on HOTTOPIC	$e^{\Omega(n)}$ on HOTTOPIC	Remarks
$(1+1)$-EA	$c < 1$[a] [22]	$c < c_0$	$c > c_0$[22]	[a]runtime $O(n \log^2 n)$ for all $c \leq 1 + \varepsilon$ [21]
$(1+\lambda)$-EA	$c < 1$	$c < c_0$	$c > c_0$	
$(\mu+1)$-EA	??	$c < c_0$	$c > c_0$	
$(\mu+1)$-GA	??	c arbitrary[b]	only if μ too small	[b]for $\mu = \mu(c)$ large enough
$(1+(\lambda,\lambda))$-GA	$c\gamma < 1$[c]	$c\gamma < c_0$ [c,d]	$c\gamma > c_0$	[c]holds also if c, γ, λ depend on n and/or are adaptive [d]achieves ONEMAX runtimes $\approx n\sqrt{\log n}$ and $O(n)$ for optimal [6], [8] static and adaptive parameters, resp.
$(1+1)$-fEA	$m_2/m_1 < 1$	$m_2/m_1 < 1$ $\Phi < 1$[f]	$m_2/m_1 > 1$[e] $\Phi > 1$[f]	[e]only if $\Pr[\mathcal{D}=1]$ is small enough. [f]Φ is similar to m_2/m_1, but has correction term for $\Pr[\mathcal{D}=1]$, see (17) on page 9.
$(1+\lambda)$-fEA	$m_2/m_1 < 1$[g]	$m_2/m_1 < 1$ $\Phi < 1$[i]	$m_2/m_1 > 1$[h] $\Phi > 1$ any power law, exp. < 2 $\Pr[\mathcal{D}=1] < 4/9 \Pr[\mathcal{D}=3]$	[g]if starting point is at most εn from optimum. [h]only if $\Pr[\mathcal{D}=1]$ is small enough. [i]if $\Pr[\mathcal{D}=1] = \Omega(1)$.
$(\mu+1)$-fEA	??	$m_2/m_1 < 1$[j] $\Phi < 1$[j]	$m_2/m_1 > 1$[j,k] $\Phi > 1$[j] any power law, exp. < 2 [j] $\Pr[\mathcal{D}=1] < 4/9 \Pr[\mathcal{D}=3]$ [j]	[j]if $\Pr[\mathcal{D}=0] = \Omega(1)$. [k]only if $\Pr[\mathcal{D}=1]$ is small enough.
$(\mu+1)$-fGA	??	$\mathcal{D}$ arbitrary[ℓ]	only if μ too small	[ℓ]for $\mu = \mu(\mathcal{D})$ large enough, if $\Pr[\mathcal{D}=0] = \Omega(1)$.

least one such mutation is present in the best mutant, and is thus available for crossover. In the most relevant regime, where the expected number of flipped zero-bits in *any* mutation is small (say, at most one), the probability to be selected increases by a factor of $\Theta(\lambda)$ [from $1/\lambda$ to $\Theta(1)$] if a zero-bit is flipped. For monotone functions we do show that the probability of being selected can only increase with the number of flipped zero-bits. However, there is no apparent reason that it should increase by a factor of $\Theta(\lambda)$, or by any significant factor at all. In fact, it is not hard to see that for the linear function BINVAL it only increases by a constant factor.

4) Fast $(1+1)$-EA, Fast $(1+\lambda)$-EA, Fast $(\mu+1)$-EA: These algorithms, which we abbreviate by $(1+1)$-fEA, $(1+\lambda)$-fEA, and $(\mu+1)$-fEA have recently been proposed by Doerr *et al.* [12], and they have immediately attracted considerable attention (e.g., [14], [15], and [23]). The idea is to replace the standard bit mutation, in which each bit is flipped independently, by a *heavy-tailed* distribution $\mathcal{D}$. That is, in each round we draw a number s from some heavy-tailed distribution (for example, a *power-law distribution* with $\Pr[s=k] \sim k^{-\kappa}$ for some $\kappa > 1$, also called *Zipf distribution*). Then, the mutant is generated from the parent by flipping exactly s bits, chosen uniformly at random. In this way, most mutations are generated by flipping only a small number of bits, but there is a substantially increased probability to flip many bits. This approach has given some hope to unify the best of the two worlds: 1) small mutation rate and 2) large mutation rate.

For monotone functions, our results are rather discouraging. This is not completely unexpected since the algorithms build on the very idea of increasing the probability of large mutation rates. We show a dichotomy for the $(1+1)$-fEA with respect to m_2/m_1, where $m_1 := \mathbb{E}[s]$ and $m_2 := \mathbb{E}[s(s-1)]$ are the first and second falling moment of the distribution $\mathcal{D}$, although the results are subject to some technical conditions.[5] As before, if $m_2/m_1 < 1$ then the runtime is $O(n \log n)$ for all monotone functions. On the other hand, if $m_2/m_1 > 1$ and additionally $p_1 := \Pr[\mathcal{D}=1]$ is sufficiently small then the runtime on some HOTTOPIC instances is exponential. As for the other algorithms, we get a sharp threshold for the parameter regime that is efficient on HOTTOPIC, so we can decide for each distribution whether it leads to fast or to exponential runtimes on HOTTOPIC. Due to a correction term related to p_1 (17), it *is* possible to construct heavy-tail distributions which are efficient on all HOTTOPIC functions, but they must be chosen with great care. For example, no power-law distribution with exponent $\kappa \in (1,2)$ is efficient, which includes the choice $\kappa = 1.5$ that is used for experiments in [12] and [23]. Also, no distribution with $p_1 < (4/9)\Pr[\mathcal{D}=3]$ is efficient on HOTTOPIC. In general, our findings contrast the results in [12], where larger tails (smaller κ) lead to faster runtimes.

As before, larger values of λ and μ do not seem to have any influence as long as crossover is not allowed. For the $(1+\lambda)$-fEA and $(\mu+1)$-fEA, we show exactly the same results as for the $(1+1)$-fEA, except that we could not show runtime bounds for *all* monotone functions if $m_2/m_1 < 1$. Rather, we only show them for HOTTOPIC. Thus, we cannot exclude the possibility that larger values of λ, μ make things even worse.

5) Fast $(\mu+1)$-GA: As for the classical algorithms, crossover tremendously improves the situation. For every distribution $\mathcal{D}$ with $\Pr[\mathcal{D}=1] = \Omega(1)$, if μ is a sufficiently large constant then the $(\mu+1)$-fGA optimizes HOTTOPIC in time $O(n \log n)$. As for the $(\mu+1)$-GA, it is an open question whether the same result carries over to *all* monotone functions.

Further Results: For all algorithms, the regime of exponential runtime does not just mean that it is hard to find the optimum, but rather the algorithms do not even come close.

[5]Note that a heavy tail generally increases m_2 much stronger than m_1, so it increases the quotient m_2/m_1.

More precisely, in all these cases there is an $\varepsilon > 0$ (depending only on c or on the other dichotomy parameters) such that the probability that any of the EAs or GAs finds a search point with at least $(1-\varepsilon)n$ correct bits within a subexponential time is exponentially small as $n \to \infty$. The size of ε can be quite considerable if the parameter c is much larger than $c_0 = 2.13\ldots$ For example, simulations suggest for the (1+1)-EA that $\varepsilon \approx 0.15$ for $c = 4$ and $\varepsilon \approx 0.09$ for $c = 3$.[6] On the other hand, starting close to the optimum does not help either: for every $\varepsilon > 0$ there are monotone function such that if the EAs or GAs are initialized with random search points with εn incorrect bits, then still the algorithms need exponential time to find the optimum.

Summary: It appears that increasing the number of offspring λ or the population size μ does not help at all to overcome the detrimental effects of large mutation rate in EAs. All EAs are highly vulnerable even to a very moderate increase of the mutation rate. This puts the HOTTOPIC functions in sharp contrast to other benchmarks like TSP, SAT, and mixed integer programs, in which parameters have a much more moderate effect on the performance [28].[7] Using heavy tails as in the fEAs seems to make things even worse, although the picture gets more complicated. On the other hand, using crossover can remedy the effect of large mutation rates, and can extend the range of good mutation rates arbitrarily.

B. Intuition on HotTopic

Now, we give an intuition why the HOTTOPIC functions are hard to optimize for large mutation rates. Note that a monotone function, by its very definition, has a local "gradient" that always points into the same corner of the hypercube, in the sense that for each bit individually, in all situations we prefer a one-bit over a zero-bit. The construction by Lengler and Steger [22] distorts the gradient by assigning different positive weights to the components. Such a distortion cannot alter the direction of the gradient by too much. In particular, following the gradient will always decrease the distance from the optimum. This is why algorithms with small mutation rate may find the optimum; they follow the gradient relatively closely. However, the weights in [22] are chosen such that there is always a hot topic, i.e., a subdirection of the gradient which is highly preferred over all other directions. Focusing too much on this hot topic will lead to a behavior that is very good at optimizing this particular aspect—but all other aspects will deteriorate a little because they are out of focus. Thus, if the hot topic is sufficiently narrow and changes often, then advances in this aspect will be overcompensated by a decline in the neglected parts, which leads overall to stagnation.

This last sentence is not merely a pessimistic allegory on scientific progress, but it also happens for EAs with large mutation rates. They will put the currently preferred direction above everything else, and will accept any mutation that makes progress in that direction, regardless of the harm that such a mutation may cause on other bits. This may lead in total to a drift away from the optimum, since random walk steps naturally tend to increase the distance from the optimum. For the fEAs or fGAs, this effect is amplified if the algorithm is close to the optimum. In this case, the probability to find any improvement at all is very small, and we typically find an improvement in an aggressive step in which many bits are flipped. Then, the same step also typically causes a lot of errors among the low-priority bits. For the same reason, an adaptive choice of the mutation strength c may be harmful if it increases the mutation parameter in phases of stagnation: close to the optimum, most steps are stagnating, and increasing the mutation parameter indeed increases the probability to find a better search point in the hot topic direction (though not the probability to make *any* improvement). This may fatally lead to a large mutation parameter. We leave the rigorous study of adaptive parameter choices to future work.

II. Preliminaries and Definitions

A. Notation

Throughout this paper, we will assume that $f : \{0,1\}^n \to \mathbb{R}$ is a monotone function, i.e., for every $x, y \in \{0,1\}^n$ with $x \neq y$ and such that $x_i \geq y_i$ for all $1 \leq i \leq n$ it holds $f(x) > f(y)$.[8] We will consider algorithms that try to maximize f, and we will mostly focus on the *runtime* of an algorithm, which we define as the number of function evaluations until the first evaluation of the global maximum of f.

We say that an EA or GA is *elitist* [13] if the selection operator greedily chooses the fittest individuals to form the next generation. We call an EA or GA *unbiased*, [19] if the mutation and crossover operators are invariant under the isomorphisms of $\{0,1\}^n$, i.e., if mutation and crossover are symmetric with respect to the ordering of the bits, and with respect to exchange of the values 0 and 1. All algorithms considered in this paper are unbiased.

For $n \in \mathbb{N}$, we denote $[n] := \{1, \ldots, n\}$. We use the notation $x = y \pm z$ to abbreviate $x \in [y-z, y+z]$. For a search point x, we write $\textsc{Om}(x)$ for the ONEMAX potential, i.e., the number of one-bits in x. For $x \in \{0,1\}^n$ and $\emptyset \neq I \subseteq [n]$, we denote by $d(I,x) := |\{i \in I | x_i = 0\}|/|I|$ the *density* of zero bits in I. In particular, $d([n], x) = 1 - \textsc{Om}(x)/n$.

All Landau notation $O(n), o(n), \ldots$ is with respect to $n \to \infty$. For example, $\lambda = O(1)$ means that there is a constant $C > 0$, independent of n, such that $\lambda = \lambda(n) \leq C$ for all $n \in \mathbb{N}$. We say that an event $\mathcal{E} = \mathcal{E}(n)$ holds *with high probability* or *whp* if $\Pr[\mathcal{E}(n)] \to 1$ for $n \to \infty$. We say that $\mathcal{E}(n)$ is *exponentially unlikely* if $\Pr[\mathcal{E}(n)] = e^{-\Omega(n)}$, and that is *exponentially likely* if $\Pr[\mathcal{E}(n)] = 1 - e^{-\Omega(n)}$.

For an event $\mathcal{E}$, we let $\mathbb{I}[\mathcal{E}]$ be the indicator variable which is one if $\mathcal{E}$ occurs, and zero otherwise. For a distribution $\mathcal{D}$, by abuse of notation write $\Pr[\mathcal{D} = x]$ for $\Pr[X = x | X \sim \mathcal{D}]$.

Throughout this paper, we will be slightly sloppy about conditional probabilities $\Pr[A|B]$ and expectation, and we will ignore cases in which $\Pr[B] = 0$. We use the term *increasing function* as equivalent to the term *nondecreasing function*, and likewise for *decreasing function*. The only exception is for the term *monotone*, as monotone functions are automatically assumed to be strictly monotone.

[6] Details can be found in the online supplementary material.

[7] The configuration landscape could still be unimodal as in [28], but for exponential regimes this question is far less important.

[8] Note that this property might more correctly be called *strictly monotone*, but in this paper we will stick with the term *monotone* for brevity.

Finally, throughout this paper, we will use n for the dimension of the search space, μ and λ for the population size and offspring population size, respectively, c for the mutation parameter, γ for the crossover parameter of the $(1+(\lambda,\lambda))$-GA, and $\mathcal{D}, m_1, m_2$ for the bit flip distribution of the fast EAs and GAs and its first and second falling moment $\mathbb{E}[s|s\sim\mathcal{D}]$ and $\mathbb{E}[s(s-1)|s\sim\mathcal{D}]$, respectively. Unless otherwise stated, we will assume that $\mu, \lambda, c, \gamma = \Theta(1)$ and $m_1 = \Omega(1)$.

B. Algorithms

Most algorithms that we consider fall into the class of $(\mu+\lambda)$ EAs, $(\mu+\lambda)$-EAs, or $(\mu+\lambda)$ genetic algorithms, $(\mu+\lambda)$-GAs. They can be described by the framework in Algorithm 1. In a nutshell, they maintain a population of size μ. In each *generation*, λ additional offspring are created by *mutation* and possibly *crossover*, and the μ search points of highest fitness among the $\mu+\lambda$ individuals form the next generation. Thus, we use an *elitist selection* scheme. In EAs, the offspring are only created by *mutation*, in GAs they are either created by mutation or by crossover. For mutation we use standard bit mutation as a default, in which each bit is independently flipped with probability c/n, where c is the *mutation parameter*. The only exception are the *fast* EAs and GAs, in which first the number s of bit mutations is drawn from some distribution $\mathcal{D} = \mathcal{D}(n)$, and then exactly s bits are flipped, chosen uniformly at random. In particular, we may write $m_1 = \mathbb{E}[s]$ and $m_2 = \mathbb{E}[s(s-1)]$, since we defined m_1 and m_2 to be the first and second falling moment of $\mathcal{D}$, respectively. We will always assume that $\mu, \lambda, c = \Theta(1)$.[9]

An exception to the above scheme is the $(1+(\lambda,\lambda))$-GA [7]. Here, the population consists of a single search point x. Then in each round, we pick $s \sim \text{BIN}(n, c/n)$, and create λ offspring from x by flipping exactly s bits in x uniformly at random. Then, we select the fittest offspring y among them, and we perform λ independent biased crossover between x and y, where for each bit we take the parent gene from y with probability γ, and the gene from x otherwise. If the best of these crossover offspring is at least as fit as x, then it replaces x. We will usually assume that $\lambda, c, \gamma = \Theta(1)$, unless otherwise mentioned.

C. Hard Monotone Functions: HotTopic

In this section, we give the construction of hard monotone functions by Lengler and Steger [22], following closely their exposition. The functions come with four parameters α, β, ρ, and ε, and they are given by a randomized construction. We call the corresponding function $\text{HOTTOPIC}_{\alpha,\beta,\rho,\varepsilon} = \text{HT}_{\alpha,\beta,\rho,\varepsilon} = \text{HT}$. The hard regime of the parameters is

$$1 > \alpha \gg \varepsilon \gg \beta \gg \rho > 0 \tag{1}$$

by which we mean that $\alpha \in (0,1)$ is a constant, $\varepsilon = \varepsilon(\alpha)$ is a sufficiently small constant, $\beta = \beta(\alpha,\varepsilon)$ is a sufficiently small constant, and $\rho = \rho(\alpha,\varepsilon,\beta)$ is a sufficiently small constant.

[9]There are many variants of the algorithms that we use here. For example, for GAs it is not important that the probability for crossover is exactly $1/2$. In fact, it is also common in GAs to create each offspring by a crossover and a mutation. The proofs and results of this paper carry over to these variants.

Algorithm 1: $(\mu+\lambda)$-EA or $(\mu+\lambda)$-GA With Mutation Parameter c for Maximizing an Unknown Fitness Function $f : \{0,1\}^n \to \mathbb{R}$. The EA-Algorithms Do Not Crossover. X Is a Multiset, i.e., It May Contain Search Points Several Times

Initialization:
  $X \leftarrow \emptyset$;
  for $i = 1, \dots, \mu$ **do**
    Sample $x^{(i)}$ uniformly at random from $\{0,1\}^n$;
    $X \leftarrow X \cup \{x^{(i)}\}$;
Optimization: for $t = 1, 2, 3, \dots$ **do**
  for $i = 1, 2, \dots \lambda$ **do**
    For GA, flip a fair coin to do either a mutation or a crossover; for EA, always do a mutation.
    if *mutation* **then**
      Choose $x \in X$ uniformly at random;
      Create $y^{(j)}$ by flipping each bit in x independently with probability c/n;
    if *crossover* **then**
      Choose $x, x' \in X$ independently uniformly at random;
      Create $y^{(j)}$ by setting $y_i^{(j)}$ to either x_i or x'_i, each with probability $1/2$, independently for all bits;
  Selection:
  Set $X \leftarrow X \cup \{y^{(1)}, \dots, y^{(\lambda)}\}$;
  for $i = 1, \dots, \lambda$ **do**
    Select $x \in \arg\min\{f(x)|x \in X\}$ (break ties randomly) and update $X \leftarrow X \setminus \{x\}$;

Now, we come to the construction. For $1 \le i \le e^{\rho n}$ we choose sets $A_i \subseteq [n]$ of size αn independently and uniformly at random, and we choose subsets $B_i \subseteq A_i$ of size βn uniformly at random. We define the *level* $\ell(x)$ of a search point $x \in \{0,1\}^n$ by

$$\ell(x) := \max\{\ell' \in [e^{\rho n}] : |\{j \in B_{\ell'} : x_j = 0\}| \le \varepsilon\beta n\} \tag{2}$$

where we set $\ell(x) = 0$, if no such ℓ' exists). Then, we define $f : \{0,1\}^n \to \mathbb{R}$ as follows:

$$\text{HT}(x) := \ell(x)\cdot n^2 + \sum_{i\in A_{\ell(x)+1}} x_i \cdot n + \sum_{i\notin A_{\ell(x)+1}} x_i \tag{3}$$

where for $\ell = e^{\rho n}$ we set $A_{\ell+1} := B_{\ell+1} := \emptyset$.

So the set $A_{\ell+1}$ defines the hot topic while the algorithm is at level ℓ, where the level is determined by the sets B_i. It was shown in [22] that whp[10] the (1+1)-EA with mutation parameter $c > c_0$ needs exponential time to find the optimum.

One easily checks that this function is monotone. Indeed, assume that x is dominated by y, i.e., $x_i \le y_i$ for all $1 \le i \le n$. Then, $\ell(x) \le \ell(y)$. If $\ell(x) = \ell(y)$ then it is obvious that $f(x) \le f(y)$, and the inequality is strict if $x \ne y$. On the other hand, if $\ell(x) < \ell(y)$ then $f(x) < \ell(x)n^2 + n^2 \le \ell(y)n^2 \le f(y)$, as desired.

[10]With high probability, i.e., with probability tending to one as $n \to \infty$.

D. Tools

We will make frequent use of standard drift theorems for multiplicative drift [11], and of tail bounds for positive additive drift and for negative drift [24], [25], [29]. We use the formulation from [22]. The exact formulations can be found in the online supplementary material.

We will also repeatedly use Chebyshev's sum inequality [16], also know as rearrangement inequality.

Theorem 1 (Chebyshev's Sum Inequality): Let $(a_i)_{i\in[n]}$, $(b_i)_{i\in[n]}$ be sequences in $\mathbb{R}$, and let $(c_i)_{i\in[n]}$ be a sequence in $\mathbb{R}_0^+$ with $\sum_{i=1}^n c_i b_i > 0$.

1) If (a_n) and (b_n) are both increasing, then

$$\frac{\sum_{i=1}^n c_i a_i}{\sum_{i=1}^n c_i} \leq \frac{\sum_{i=1}^n c_i a_i b_i}{\sum_{i=1}^n c_i b_i}. \tag{4}$$

2) If (a_n) is increasing and (b_n) is decreasing, then

$$\frac{\sum_{i=1}^n c_i a_i}{\sum_{i=1}^n c_i} \geq \frac{\sum_{i=1}^n c_i a_i b_i}{\sum_{i=1}^n c_i b_i}. \tag{5}$$

The theorem also holds for infinite sequences if all sums converge.

III. Upper Bounds for General Monotone Functions

In this section, we will give a generic proof for strong dichotomies, i.e., for showing that under certain circumstances an algorithm will optimize every monotone function in time $O(n \log n)$. The proof follows loosely proofs from [10] and [22].

Theorem 2 (Generic Easiness Proof): Consider an elitist algorithm $\mathcal{A}$ with population size one that in each round generates an offspring by an arbitrary method, and replaces the parent if and only if the offspring has at least the same fitness. Let s_{01} denote the number of zero-bits in the parent that are one-bits in the offspring, and vice versa for s_{10}. Assume that there is a constant $\delta > 0$ such that for all $x \in \{0,1\}^n$

$$\mathbb{E}[s_{10} | \text{parent} = x \text{ and } s_{01} > 0] \leq 1 - \delta \tag{6}$$

and

$$\Pr[s_{01} > 0 | \text{parent} = x] = \Omega(\tfrac{1}{n}(n - \textsc{Om}(x))). \tag{7}$$

Then, $\mathcal{A}$ finds the optimum of every strictly monotone functions in $O(n \log n)$ rounds, with high probability and in expectation.

Before we prove the theorem, we remark that the $(1+\lambda)$-EA, the $(1+1)$-fEA, and the $(1+(\lambda,\lambda))$-GA all fit the generic description in Theorem 2, modulo condition (6). For the $(1+(\lambda,\lambda))$-GA, note that the procedure to generate the offspring is rather complicated, and involves several intermediate mutation and crossover steps. Nevertheless, the procedure ultimately produces a single offspring (the fittest of the crossover offspring) which competes with the parent.

Proof of Theorem 2: Let $X_t := n - \textsc{Om}(x^{(t)})$, where $x^{(t)}$ is the t-th search point of $\mathcal{A}$, and let y be the offspring of $x^{(t)}$. First note that if $s_{01} = 0$ and $x \neq y$, then by monotonicity $f(x) > f(y)$. Therefore, $x^{(t+1)} = x^{(t)}$ and $X_{t+1} = X_t$ if $s_{01} = 0$. This also holds in the trivial case $s_{01} = 0$ and $x = y$.

If $s_{01} > 0$ and $s_{10} = 0$, then again by monotonicity $f(y) > f(x)$. Thus, $x^{(t+1)} = y^{(t)}$ and $X_{t+1} \leq X_t - 1 = X_t - 1 + s_{10}$.

Finally, if $s_{01} > 0$ and $s_{10} > 0$ then we have two cases. Either $y^{(t)}$ is accepted, in which case $X_{t+1} = X_t - s_{01} + s_{10} \leq X_t - 1 + s_{10}$. Or $y^{(t)}$ is rejected, in which case the same inequality follows from $X_{t+1} = X_t \leq X_t - 1 + s_{10}$.

Summarizing, we see that X_t does not change for $s_{01} = 0$, and that for $s_{01} > 0$ we have in all cases $X_{t+1} \leq X_t - 1 + s_{10}$. Therefore, X_t has a drift of at least

$$\mathbb{E}[X_t - X_{t+1} | x^{(t)}] \geq \Pr[s_{01} > 0] \cdot \mathbb{E}[1 - s_{10} | s_{01} > 0, x^{(t)}] \overset{(6),(7)}{=} \Omega(\delta/n \cdot X_t).$$

The claim thus follows from the multiplicative drift theorem. ∎

From, Theorem 2, it will follow that the $(1+\lambda)$-EA with $c < 1$, the $(1+1)$-fEA with $m_2/m_1 < 1$, and the $(1+(\lambda,\lambda))$-GA with $c\gamma < 1$ have runtime $O(n \log n)$, since we will show that these settings satisfy (6). For the $(1+(\lambda,\lambda))$-GA with $c\gamma < 1$ and nonconstant parameters we cannot apply Theorem 2 directly. However, we will see that the conditional expectation in (6) is still the crucial object to study.

Theorem 3: Let $\delta > 0$. For any strictly monotone function, with high probability the following algorithms find the optimum in $O(n \log n)$ generations.

1) The $(1+\lambda)$-EA with $c \leq 1 - \delta$ and $c = \Omega(1)$.
2) The $(1+1)$-fEA with $m_2/m_1 \leq 1 - \delta$ and $m_1 = \Omega(1)$.
3) The $(1+(\lambda,\lambda))$-GA with $c\gamma \leq 1 - \delta$ and $c\gamma = \Omega(1)$.

Moreover, if the $(1+(\lambda,\lambda))$-GA with $c\gamma < 1 - \delta$ uses the optimal static or adaptive parameter choice from [6],[11] then with high probability the runtime on HotTopic is up to a factor $\Theta(1)$ the same as the runtime for OneMax.

We remark that the optimal runtime of the $(1+(\lambda,\lambda))$-GA on OneMax is $O(n\sqrt{\log(n)\log\log\log(n)/\log\log n})$ for static parameters and $O(n)$ for adaptive parameter choices [6].

Proof of Theorem 3: First consider the $(1+\lambda)$-EA. Assume that the current search point is x. We create the λ offspring by two consecutive steps. For each $j \in [\lambda]$, first we flip every zero-bit in x independently with probability c/n, and call the result $z^{(j)}$. Then, for every one-bit in x, we flip the corresponding bit in $z^{(j)}$ independently with probability c/n, and call the result $y^{(j)}$. Thus, $y^{(j)}$ follows exactly the right distribution: each bit has been flipped independently with probability c/n. Let $k \in [\lambda]$ be the random variable that denotes the index of the fittest of the $y^{(j)}$ (breaking ties randomly). Moreover, fix some index $i \in [n]$ for which $x_i = 1$.

Note that for every fixed j, we have $\Pr[y_i^{(j)} = 0] = c/n$. Intuitively, we need to show that the probability does not increase by the selection process. For all $j \in [\lambda]$, let $p_j := \Pr[k = j | z^{(1)}, \ldots, z^{(\lambda)}]$. By monotonicity, if a search point $y^{(j)}$ with $y_i^{(j)} = 0$ is the fittest of the offspring, then replacing $y_i^{(j)} = 0$ by $y_i^{(j)} = 1$ can only increase the fitness. Therefore, conditioning on $y_i^{(j)} = 0$ can only decrease the probability that

[11] In fact, the suggested parameter choice in [6] and [8] satisfies $c\gamma = 1$ instead of $c\gamma < 1$. However, the runtime analysis in [8] only changes by constant factors if γ is decreased by constant factors. Thus, Theorem 3 applies to the parameter choices from [6] and [8], except that γ is smaller by a constant factor.

$k = j$, in formula

$$\Pr\Big[k = j \text{ and } y_i^{(j)} = 0 \,|\, z^{(1)}, \ldots, z^{(\lambda)}\Big] \le p_j \cdot \Pr\Big[y_i^{(j)} = 0\Big] \quad (8)$$

where we note that the latter probability is independent of $z^{(1)}, \ldots, z^{(\lambda)}$. Let $s_{01}^{(j)}$ be the number of zero-bits in which x and $z^{(j)}$ differ, i.e., the number of bits that have been flipped from zero to one. Let $J \coloneqq \{j \in [\lambda] | s_{01}^{(j)} > 0\}$. Then, by (8)

$$\Pr\Big[y_i^{(k)} = 0 | s_{01}^{(k)} > 0; \quad z^{(1)}, \ldots, z^{(\lambda)}\Big] \le \frac{\sum_{j\in J} p_j \cdot \Pr\Big[y_i^{(j)} = 0\Big]}{\sum_{j\in J} p_j} = c/n.$$

Summing over all $i \in [n]$ with $x_i = 1$, and averaging over all possible values of $z^{(1)}, \ldots, z^{(\lambda)}$, we obtain

$$\mathbb{E}\Big[|\Big\{i \in [n] : x_i = 1, y_i^{(k)} = 0\Big\}| | s_{01}^{(k)} > 0\Big] \le c \le 1 - \delta. \quad (9)$$

Thus, condition (6) in Theorem 2 is satisfied. Note that so far we have not used $c = \Omega(1)$. We only need this assumption to verify condition (7), which indeed follows immediately. So the statement for the $(1 + \lambda)$-EA follows from Theorem 2.

Next we turn to the $(1 + 1)$-fEA. As before, let x be the current search point, and let y be the offspring. For all $s > 0$, let p_s be the probability to flip exactly s bits. Then, we have

$$\begin{aligned} 2 \cdot \Pr[1 \le s \le 2] &\ge m_1 - \sum_{s=3}^{\infty} p_s s \ge m_1 - \frac{1}{2}\sum_{s=3}^{\infty} p_s s(s-1) \\ &\ge m_1 - \frac{m_2}{2} \ge \frac{m_1}{2} \end{aligned} \quad (10)$$

so $\Pr[1 \le s \le 2] \ge m_1/4$. In particular, this expression is in $\Omega(1)$, which implies condition (7) in Theorem 2. Note for later reference that (10) also implies $m_1 \le 4$.

It remains to check (6). For this, let s_{01} and s_{10} denote the number of bit flips from 0 to 1 and from 1 to 0, respectively. We observe that

$$\begin{aligned} \mathbb{E}[s_{10}|s_{01} > 0] &= \frac{\sum_{s\ge 1} p_s \Pr[s_{01} > 0|s]\, \mathbb{E}[s_{10}|s; s_{01} > 0]}{\sum_{s\ge 1} p_s \Pr[s_{01} > 0|s]} \\ &\le \frac{\sum_{s\ge 1} p_s \Pr[s_{01} > 0|s] \cdot (s-1)}{\sum_{s\ge 1} p_s \Pr[s_{01} > 0|s]}. \end{aligned} \quad (11)$$

To estimate the term above, we note that since the term $s - 1$ is increasing, for every nondecreasing sequence α_s, by Chebyshev's sum inequality we may bound

$$(11) \le \frac{\sum_{s\ge 1} p_s \Pr[s_{01} > 0|s] \cdot \alpha_s \cdot (s-1)}{\sum_{s\ge 1} p_s \Pr[s_{01} > 0|s] \cdot \alpha_s}. \quad (12)$$

We will use $\alpha_s \coloneqq s/\Pr[s_{01} > 0|s]$, so we need to show that $\alpha_s^{-1} = \Pr[s_{01} > 0|s]/s$ is a nonincreasing sequence. We regard the process where we draw the s bit positions one after another, and consider for the ith round the probability q_i that a zero bit is drawn for the first time in this round. This probability is decreasing, and thus

$$\begin{aligned} \frac{1}{s}\Pr[s_{01} > 0|s] &= \frac{1}{s}\sum_{i=1}^{s} q_i \le \frac{1}{s-1}\sum_{i=1}^{s-1} q_i \\ &= \frac{1}{s-1}\Pr[s_{01} > 0|s-1] \end{aligned}$$

as desired. Plugging α_s into (12) yields

$$\mathbb{E}[s_{10}|s_{01} > 0] \le \frac{m_2}{m_1} \le 1 - \delta \quad (13)$$

so condition (6) in Theorem 2 is satisfied, and the statement follows from Theorem 2. For later reference, we note that the first inequality in (13) holds for any distribution $\mathcal{D}$, regardless whether $(m_2/m_1) \le 1 - \delta$.

Finally, let us turn to the $(1 + (\lambda, \lambda))$-GA. Let x be the current search point. In the first step an integer $s \sim \text{BIN}(n, c/n)$ is chosen, and λ offspring $y^{(1)}, \ldots, y^{(\lambda)}$ are created from x by flipping exactly s bits. As for the $(1 + \lambda)$-EA, let $s_{01}^{(j)}$ and $s_{10}^{(j)} = s - s_{01}^{(j)}$ be the number of zero-bits and one-bits that were flipped in the creation of $y^{(j)}$, respectively, and let $k \in [\lambda]$ be the fittest among the $y^{(j)}$, breaking ties randomly. Note that for a fixed j, the offspring $y^{(j)}$ has the same distribution as for the $(1 + \lambda)$-EA. (The difference is that the offspring are not independent.) Therefore, for every fixed $j \in [\lambda]$ and all $r, r' \in \mathbb{N}$

$$\Pr\Big[s_{10}^{(j)} \ge r | s_{01}^{(j)} \ge r'\Big] = \Pr\big[\text{BIN}(\text{OM}(x), c/n) \ge r\big]. \quad (14)$$

In particular, $\mathbb{E}[s_{10}^{(j)} | s_{01}^{(j)} \ge r'] \le c$.

Now, we show that $\mathbb{E}[s_{10}^{(j)} | s_{01}^{(j)} \ge r']$ can only decrease by the selection process. Fix any values of $s, s_{01}^{(1)}, \ldots, s_{01}^{(\lambda)}$, and let $p_j \coloneqq \Pr[k = j | s, s_{01}^{(1)}, \ldots, s_{01}^{(\lambda)}]$. Thus, we condition on the number of zero-bits and one-bits that we flip, but not on their location. Note that for fixed s we can create a random offspring with $s_{01}^{(j)} = \sigma$ (i.e., with σ flips of zero-bits and $s - \sigma$ flips of one-bits of x) by starting with a random search point with $s_{01}^{(j)} = \sigma - 1$, reverting a random flip of a one-bit, and adding a random flip of a zero-bit of x. Since this operation strictly increases the fitness, it can only increase p_j. Hence, p_j is an increasing function in $s_{01}^{(j)}$. By symmetry of the selection operator, this implies in particular that $p_i \le p_j$ if $s_{01}^{(i)} \le s_{01}^{(j)}$ for all indices $i, j \in [\lambda]$. On the other hand, the indicator function $\mathbb{I}[x \le s - r]$ is trivially decreasing in x. Therefore, using the index set $J \coloneqq \{j \in [\lambda] | s_{01}^{(j)} \ge r'\}$, the sequences $a_j \coloneqq \mathbb{I}[s_{10}^{(j)} \ge r]$ and $b_j \coloneqq p_j$, where $j \in J$, are oppositely sorted. So we may use Chebyshev's sum inequality (5) (with $c_j \coloneqq 1$ for all $j \in J$), and obtain for any $r \in \mathbb{N}$

$$\begin{aligned} &\Pr\Big[s_{10}^{(k)} \ge r | s, s_{01}^{(1)}, \ldots, s_{01}^{(\lambda)} \text{ and } s_{01}^{(k)} \ge r'\Big] \\ &= \frac{\sum_{j\in J} p_j \cdot \mathbb{I}\Big[s_{10}^{(j)} \ge r\Big]}{\sum_{j\in J} p_j} = \frac{\sum_{j\in J} p_j \cdot \mathbb{I}\Big[s_{01}^{(j)} \le s - r\Big]}{\sum_{j\in J} p_j} \\ &\le \frac{1}{|J|}\sum_{j\in J} \mathbb{I}\Big[s_{01}^{(j)} \le s - r\Big] = \frac{1}{|J|}\sum_{j\in J} \mathbb{I}\Big[s_{10}^{(j)} \ge r\Big]. \end{aligned}$$

Note that the latter term just counts which fraction of those j with $s_{01}^{(j)} \ge r'$ also satisfy $s_{10}^{(j)} \ge r$. This is directly related to the definition of conditional probability. In particular, averaging over all possible values of $s, s_{01}^{(1)}, \ldots, s_{01}^{(\lambda)}$, we get for every fixed $j \in [\lambda]$

$$\Pr\Big[s_{10}^{(k)} \ge r | s_{01}^{(k)} \ge r'\Big] \le \Pr\Big[s_{10}^{(j)} \ge r | s_{01}^{(j)} \ge r'\Big]. \quad (15)$$

In other words, $s_{10}^{(k)}$ is stochastically dominated by $s_{10}^{(j)}$ if we condition on $s_{01} \ge r'$. Recall that the latter one is a binomial

distribution by (14), and in particular $\mathbb{E}[s_{10}^{(k)} | s_{01}^{(k)} \geq r'] \leq c$. By an analogous argument, the selection process can only increase how many zero-bits of x are flipped, i.e., $s_{01}^{(k)}$ stochastically dominates $s_{01}^{(j)}$ if we condition on $s_{01} \geq r'$.

In the second step of the $(1 + (\lambda, \lambda))$-GA, the algorithm produces λ biased crossovers $z^{(1)}, \ldots, z^{(\lambda)}$ between x and $y^{(k)}$, choosing the bits from $y^{(k)}$ with probability γ. Then, it compares the fittest crossover offspring $z^{(\ell)}$ with x. Similarly as before, we let $t_{01}^{(j)}$ be the number of bits that are zero in x and one in $z^{(j)}$, and vice versa for $t_{10}^{(j)}$. To estimate $\mathbb{E}[t_{10}^{(j)} | t_{01}^{(j)} > 0]$, we define the following three terms:

$$A_\sigma := \Pr\left[t_{01}^{(j)} > 0 | s_{01}^{(k)} = \sigma\right] / \Pr\left[t_{01}^{(j)} > 0\right]$$
$$B_\sigma := \Pr\left[s_{01}^{(k)} = \sigma\right]$$
$$C_\sigma := \mathbb{E}\left[t_{10}^{(j)} | t_{01}^{(j)} > 0 \text{ and } s_{01}^{(k)} = \sigma\right].$$

We observe that A_σ is increasing in σ with $A_0 = 0$, and that $\sum_{\sigma=0}^{\infty} B_\sigma = \sum_{\sigma=0}^{\infty} A_\sigma B_\sigma = 1$. For convenience, we also set $A_{-1} := 0$. Moreover, observe that conditioned on $s_{01}^{(k)} = \sigma$, the term $t_{10}^{(j)}$ is independent of the event $t_{01}^{(j)} > 0$, since the crossover treats bits independently. Thus, we may equivalently write $C_\sigma = \mathbb{E}[t_{10}^{(j)} | s_{01}^{(k)} = \sigma]$. Consequently, for every $\sigma \geq 0$, using (15) and the sentences thereafter in the last step

$$\begin{aligned}\sum_{r=\sigma}^{\infty} B_r C_r &= \Pr\left[s_{01}^{(k)} \geq \sigma\right] \mathbb{E}\left[t_{10}^{(j)} | s_{01}^{(k)} \geq \sigma\right] \\ &= \Pr\left[s_{01}^{(k)} \geq \sigma\right] \cdot \gamma\, \mathbb{E}\left[s_{10}^{(k)} | s_{01}^{(k)} \geq \sigma\right] \\ &\leq c\gamma \Pr\left[s_{01}^{(k)} \geq \sigma\right] = c\gamma \sum_{r=\sigma}^{\infty} B_r.\end{aligned}$$

Using summation by parts (discrete partial integration) on the two functions $g_1(\sigma) = A_\sigma$ and $g_2(\sigma) = \sum_{r=\sigma}^{\infty} B_r C_r$ [and backwards for $\tilde{g}_2(\sigma) = \sum_{r=\sigma}^{\infty} B_r$], we thus conclude that

$$\begin{aligned}\sum_{\sigma=0}^{\infty} A_\sigma B_\sigma C_\sigma &= \sum_{\sigma=0}^{\infty} \underbrace{(A_\sigma - A_{\sigma-1})}_{\geq 0} \underbrace{\sum_{r=\sigma}^{\infty} B_r C_r}_{\leq c\gamma \sum B_r} \\ &\leq c\gamma \sum_{\sigma=0}^{\infty} (A_\sigma - A_{\sigma-1}) \sum_{r=\sigma}^{\infty} B_r \\ &= c\gamma \sum_{\sigma=0}^{\infty} A_\sigma B_\sigma = c\gamma.\end{aligned}$$

Indeed we have computed a term of interest

$$\begin{aligned}\mathbb{E}\left[t_{10}^{(j)} | t_{01}^{(j)} > 0\right] &= \sum_{\sigma=0}^{\infty} \Pr\left[s_{01}^{(k)} = \sigma | t_{01}^{(j)} > 0\right] \cdot C_\sigma \\ &= \sum_{\sigma=0}^{\infty} A_\sigma B_\sigma C_\sigma \leq c\gamma < 1 - \delta.\end{aligned}$$

It remains to show that the second selection process, picking the fittest among the $z^{(1)}, \ldots, z^{(\lambda)}$, does not increase the term $\mathbb{E}[t_{10}^{(j)} | t_{01}^{(j)} > 0]$, i.e.,

$$\mathbb{E}\left[t_{10}^{(\ell)} | t_{01}^{(\ell)} > 0\right] \leq \mathbb{E}\left[t_{10}^{(j)} | t_{01}^{(j)} > 0\right] \leq c\gamma \leq 1 - \delta.$$

The argument is again the same as before: it suffices to observe that the probability that $z^{(j)}$ is the fittest crossover offspring is decreasing in $t_{01}^{(j)}$, and the claim follows from Chebyshev's sum inequality. We skip the details. This proves condition (6) in Theorem 2. For condition (7), we fix any $j \in [\lambda]$, and note that $\mathbb{E}[s_{01}^{(j)}] = (c/n)(n - \textsc{Om}(x))$. Since the probability to select $y^{(j)}$ is increasing in $s_{01}^{(j)}$, we have $\mathbb{E}[s_{01}^{(k)}] \geq \mathbb{E}[s_{01}^{(j)}] = (c/n)(n - \textsc{Om}(x))$. In particular, each crossover mutation satisfies

$$\mathbb{E}\left[t_{01}^{(j)}\right] = \gamma\, \mathbb{E}\left[s_{01}^{(k)}\right] \geq \frac{c\gamma}{n}(n - \textsc{Om}(x)).$$

As before, the probability to select $z^{(j)}$ in the second selection step is increasing in $t_{01}^{(j)}$. Thus, $\mathbb{E}[t_{01}^{(\ell)}] \geq (c\gamma/n)(n - \textsc{Om}(x))$, and condition (7) follows since $c\gamma = \Omega(1)$. Note that this is the only step in the proof where we use $c\gamma = \Omega(1)$. This concludes the proof of the first statement for the $(1 + (\lambda, \lambda))$-GA.

We come to the second statement on the $(1+(\lambda, \lambda))$-GA, on HotTopic for the optimal parameter choices in [6] and [8]. The crucial observation is that by the negative drift theorem the number of zero bits will drop below $\varepsilon\beta n$ in $O(n)$ generations. In particular, in the set B_i there are at most $\varepsilon\beta n$ zero bits, which means that the level has reached its maximum. This phase needs runtime $O(\lambda n)$, since 2λ search points are created in each generation. In both the adaptive and the nonadaptive case, this number matches asymptotically the runtime bound for the $(1 + (\lambda, \lambda))$-GA on OneMax proven in [6] and [8], respectively. [For the static parameter setting we have a runtime of $\Theta(\lambda n)$ for the optimal $\lambda = \sqrt{\log n \log\log n / \log\log\log n}$, for the adaptive setting we have $\lambda \leq \sqrt{1/(\varepsilon\beta)} = O(1)$ in this region of the search space, so $O(\lambda n) = O(n)$.] Moreover, again by the negative drift theorem, once the number of zero-bits has dropped below, say, $\varepsilon\beta n/4$, whp it will not increase again to more than $\varepsilon\beta n/2$ zero-bits for $\omega(n \log n)$ rounds. So let us assume that the number of zero-bits stays below $\varepsilon\beta n/2$. Again inspecting the choice of λ in [6] and [8], we see that $\lambda = O(\sqrt{n})$ throughout the process, so whp no offspring will ever leave the range of at most $\varepsilon\beta n$ zero-bits. However, in this range the HotTopic function is up to an additive constant equal to the OneMax function, so the remaining optimization time for HotTopic and for OneMax coincides. This proves the theorem. ∎

Our next theorem gives upper bounds on the runtime of the $(1 + \lambda)$-fEA on any monotone function, provided that $m_2/m_1 < 1$, where m_1 and m_2 are the first and second falling moments of the flip number distribution $\mathcal{D}$. We need to make the assumption that the algorithm starts at most in distance εn to the optimum. It is unclear whether this assumption is necessary, or merely an artifact of our proof.

Theorem 4: Let $\delta > 0$ be a constant, let $\lambda = O(1)$, and consider the $(1 + \lambda)$-fEA with distribution $\mathcal{D} = \mathcal{D}(n)$, whose falling moments m_1 and m_2 satisfy $m_2/m_1 \leq 1 - \delta$ and $m_1 = \Omega(1)$. Then, there is $\varepsilon > 0$ such that the $(1 + \lambda)$-fEA starting with any search point with at most εn zero-bits finds the optimum of every strictly monotone functions in time $O(n \log n)$ with high probability.

Proof: We set $\varepsilon := \delta/(96C)$, where C is some constant upper bound on λ. Let x denote the current search point. Assume for now that $d([n], x) \leq 2\varepsilon$, i.e., that x contains at most $2\varepsilon n$ zero-bits. We will justify this assumption at the end

of the proof. Let $y^{(1)}, \dots, y^{(\lambda)}$ be the offspring of x, and let $k \in [\lambda]$ be the index of the fittest offspring. For any fixed $j \in [\lambda]$, let $s_{01}^{(j)}$ and $s_{10}^{(j)}$ be the number of zero-bits and one-bits that were flipped in the creation of $y^{(j)}$, respectively. Moreover, let $J := \{j \in [\lambda] | s_{01}^{(j)} > 0\}$.

Our first step is to bound $\mathbb{E}[|J||J \neq \emptyset]$. Observe that $m_1 \leq 4$ by (10). Therefore, $\mathbb{E}[s_{01}^{(j)}] \leq 8\varepsilon$ for all $j \in [\lambda]$, and $\Pr[s_{01}^{(j)} > 0] \leq 8\varepsilon$ by Markov's inequality. Since the offspring are generated independently, $|J|$ follows a binomial distribution with expectation $\mathbb{E}[|J|] \leq 8\varepsilon\lambda$, and hence $\Pr[|J| \geq r|J \neq \emptyset] \leq (8\varepsilon\lambda)^{r-1}$ for all $r \in \mathbb{N}$. Thus, since $\varepsilon \leq 1/(16\lambda)$

$$\begin{aligned}\mathbb{E}\big[|J||J \neq \emptyset\big] &\leq \sum_{r=1}^{\infty} r \cdot \Pr\big[|J| \geq r|J \neq \emptyset\big] \\ &\leq 1 + 8\varepsilon\lambda \sum_{r=2}^{\infty} r(8\varepsilon\lambda)^{r-2} \\ &\leq 1 + 8\varepsilon\lambda \sum_{r=2}^{\infty} r(1/2)^{r-2} = 1 + 48\varepsilon\lambda \leq 1 + \frac{\delta}{2}.\end{aligned}$$

Now, we are ready to bound $\mathbb{E}[s_{10}^{(k)} | s_{01}^{(k)} > 0]$. For a fixed $j \in [\lambda]$, by (13) we have $\mathbb{E}[s_{10}^{(j)} | s_{01}^{(j)} > 0] \leq 1 - \delta$. Therefore, bounding generously

$$\begin{aligned}\mathbb{E}\Big[s_{10}^{(k)} | s_{01}^{(k)} > 0\Big] &\leq \mathbb{E}\left[\sum_{j \in J} s_{10}^{(j)} | J \neq \emptyset\right] \\ &= \mathbb{E}\big[|J||J \neq \emptyset\big] \cdot \mathbb{E}\Big[s_{10}^{(j)} | s_{01}^{(j)} > 0\Big] \\ &\leq (1 + \delta/2) \cdot (1 - \delta) \leq 1 - \frac{\delta}{2}. \quad (16)\end{aligned}$$

Therefore, condition (6) from Theorem 2 is satisfied. As before, condition (7) is easy to check. So Theorem 2 would imply the statement if we would know that no search point has more than $2\varepsilon n$ zero-bits in the first $O(n \log n)$ rounds. So it only remains to show that whp this is the case. Due to space restriction, we omit the argument in the printed version. It can be found in the online supplementary material. ∎

IV. Generic Result for HotTopic

In this section, we analyze the behavior of a generic algorithm on HotTopic, which will later serve as basis for all of our results on HotTopic for concrete algorithms. The generic algorithm uses population size one, but we will show that, surprisingly, $(\mu + 1)$ algorithm can be described by the same framework.

Theorem 5 (HotTopic, Generic Runtime): Let $0 < \alpha < 1$. Consider an elitist, unbiased optimization algorithm $\mathcal{A}$ with population size one that starts with a random search point x and in each round generates an offspring y by an arbitrary (unbiased) method, and replaces the parent x by y if $\mathrm{HT}(y) > \mathrm{HT}(x)$. For equal fitness, it may decide arbitrarily whether it replaces the parent. Let s be the random variable that denotes the total number of bits in which parent and offspring differ, and note that the distribution of s may depend on the parent. For parent x, we define

$$\Phi(x) := \frac{\mathbb{E}\big[s(s-1)(1-\alpha)^{s-1}\big]}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]} - \frac{\frac{1-\alpha}{\alpha}\Pr[s=1]}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]}. \quad (17)$$

1) If there are constants $\zeta, \zeta' > 0$ such that for all $x \in \{0,1\}^n$ with at most ζn zero-bits
$$\Phi(x) \geq 1 + \zeta' \quad (18)$$
then with high probability $\mathcal{A}$ needs an exponential number of steps to find the global optimum of $\mathrm{HotTopic}_{\alpha,\beta,\rho,\varepsilon}$ with parameters β, ρ, ε as in (1).
2) If there are constants $\zeta, \zeta' > 0$ such that for all $x \in \{0,1\}^n$ with at most ζn zero-bits
$$\Phi(x) \leq 1 - \zeta' \quad (19)$$
and if moreover $\Pr[s = 1] \geq \zeta$ and $\mathbb{E}[s(s-1)] \leq 1/\zeta$ for all parents x, then with high probability $\mathcal{A}$ needs $O(n \log n)$ steps to find the global optimum of $\mathrm{HotTopic}_{\alpha,\beta,\rho,\varepsilon}$ with parameters β, ρ, ε as in (1).
3) The statements in 1) and 2) remain true for algorithms that are only unbiased conditioned on an improving step,[12] if in 2) we also have $\Pr[\text{improving step}] \geq \zeta \cdot d([n], x)$. Moreover, the statement in 2) remains true for algorithms that are only unbiased if x has more than ζn zero-bits, and possibly biased for at most ζn zero bits, if we replace (19) by the condition $\mathbb{E}[s|\mathrm{HT}(y) > \mathrm{HT}(x)] \leq 2 - \zeta$.

Finally, there is a constant $\eta = \eta(\zeta', \alpha) > 0$ independent of ζ such that 1)–3) remain true in the presence of the following adversary A. Whenever an offspring x' is created from x that satisfies $f(x') > f(x)$ then A flips a coin. With probability $1-\eta$, she does nothing. Otherwise, she draws an integer $\tau \in \mathbb{N}$ with expectation $O(1)$ and with tail bound $\Pr[\tau \geq \tau'] = e^{-\Omega(\tau')}$ and she may change up to τ bits in the current search point.

We remark that 2) and 3) require parameters as in (1), and thus do not exclude a large runtime on HotTopic for atypical parameters, e.g., for large ε.

Proof of Theorem 5: In the print version, we only sketch the main idea, which is taken from [22]. The full proof can be found in the online supplementary material. Let $\delta > 0$ be a suitable constant to be defined later. In the following, we do not hide any of the constants $\beta, \rho, \varepsilon, \delta$ in the O-notation. Consider the algorithm after t steps. We denote by $d(S, t) := d(S, x^{(t)})$ the density of zero bits in an index set S at time t.

Fix $1 < i \leq e^{\rho n}$, and consider the algorithm at level $\ell = i-1$ in the case that $\varepsilon \leq d(A_i, t), d(R_i, t) \leq \varepsilon + \delta$ for $R_i := [n] \setminus A_i$. Then, one can show that the drift $\Delta := \mathbb{E}[d(A_i, t) - d(A_i, t+1)]$ of the density of zero bits within A_i toward zero is

$$\Delta = \frac{\varepsilon\, \mathbb{E}\big[s(1-\alpha)^{s-1}\big] \pm O\big(\varepsilon^2 + \delta\big)}{n}.$$

On the other hand, the same drift $\mathbb{E}[d(R_i, t) - d(R_i, t+1)] =: -\Delta'$ within R_i can be computed via

$$\Delta' = \frac{\varepsilon \frac{\alpha}{1-\alpha}\, \mathbb{E}\big[s(s-1)(1-\alpha)^{\sigma-1}\big] - \varepsilon \Pr[s=1] + \mathrm{err}}{n}$$

[12] That is, assume that for parent x, the next search point is drawn from some distribution $\mathcal{X}$, which is not necessarily unbiased. Then, we require that there is an unbiased distribution $\mathcal{X}'$ such that $\Pr[y \in \mathcal{X} | \mathrm{HT}(y) > \mathrm{HT}(x)] = \Pr[y \in \mathcal{X}' | \mathrm{HT}(y) > \mathrm{HT}(x)]$.

with an error term err $= \pm O(\varepsilon^2 + \delta)$. Since the sizes of A_i and R_i are αn and $(1-\alpha)n$, respectively, the total drift is $\mathbb{E}[d([n],t) - d([n],t+1)] = \alpha\Delta - (1-\alpha)\Delta'$. In particular, the total drift points toward zero if and only if $\Phi' := ([(1-\alpha)\Delta']/\alpha\Delta) < 1$. We note that Φ' is identical to Φ up to error terms, which we can make arbitrarily small by choosing ε and δ appropriately.

Thus, for 2) the density of zero bits has a drift toward zero for $\varepsilon \le d(A_i,t), d(R_i,t) \le \varepsilon + \delta$. The drift for larger values of $d(A_i,t)$ and $d(R_i,t)$ is also negative by a suitable coupling argument, which implies that the density of zero bits falls significantly below ε in linear time. At this point, the maximum level will be reached, and the HOTTOPIC function degenerates to the ONEMAX function. On the other hand, for 1) the density of zero bits has a drift away from zero, and moreover we have precise control over the drift in $d(A_i,t)$ and $d(R_i,t)$. From this we can show inductively that the density of $d([n],t)$ always stays well above ε, and that the algorithm never reaches any level $\ge i+2$ directly from level i. In other words, the algorithm never skips a level. Since there are exponentially many levels, the algorithm needs an exponential time. The statement 3) follows rather easily from 1) and 2), and we omit the details. Finally, for the adversary it suffices to show that the adversary does not significantly affect the drift, since all proofs are based on drift analysis. ■

V. CONCRETE RESULTS FOR HOTTOPIC

It turns out that Theorem 5 suffices to classify the behavior on HOTTOPIC for all algorithms that we study. On the first glance, this may seem surprising, since some of them are population-based, while Theorem 5 explicitly requires population size one. Nevertheless, we will see that it implies the following theorem.

Theorem 6 (HotTopic, Concrete Results): Let $\delta > 0$. We assume that $\mu, \lambda, c = \Theta(1)$ and $\Pr[\mathcal{D}=1] = \Omega(1)$, except for the $(1+(\lambda,\lambda))$-GA, for which we replace the condition on c by $c\gamma = \Theta(1)$. Let $c_0 = 2.13692\ldots$ be the smallest constant for which the function $c_0 x - e^{-c_0(1-x)} - (x/[1-x])$ has a solution $\alpha \in [0,1]$. For all $\alpha \in (0,1)$, with high probability each of the following algorithms optimizes the function $\text{HOTTOPIC}_{\alpha,\beta,\rho,\varepsilon}$ with parameters β, ρ, ε as in (1) in time $O(n \log n)$.

1) The $(1+\lambda)$-EA with $c \le c_0 - \delta$.
2) The $(\mu+1)$-EA with $c \le c_0 - \delta$.
3) The $(\mu+1)$-GA with arbitrary $c = \Theta(1)$ if $\mu = \mu(c)$ is sufficiently large.
4) The $(1+(\lambda,\lambda))$-GA with $c\gamma \le c_0 - \delta$.
5) The $(1+\lambda)$-fEA with $m_2/m_1 \le 1-\delta$; more generally, the $(1+\lambda)$-fEA with any distribution that satisfies (19) for $s \sim \mathcal{D}$, as well as $\Pr[\mathcal{D}=1] = \Omega(1)$.[13]
6) The $(\mu+1)$-fEA with parameters as in the preceding case, if additionally $\Pr[\mathcal{D}=0] = \Omega(1)$.
7) The $(\mu+1)$-fGA with arbitrary $\mathcal{D}$ with $\Pr[\mathcal{D}=0] = \Omega(1)$, if $\mu = \mu(\mathcal{D})$ is sufficiently large.

On the other hand, for each of the following algorithms there exists $\alpha_0 \in [0,1]$ such that with high probability the algorithm needs exponential time to optimize the function $\text{HOTTOPIC}_{\alpha_0,\beta,\rho,\varepsilon}$ with parameters β, ρ, ε as in (1). In the first four cases we may choose $\alpha_0 = 0.237134\ldots$

1) The $(1+\lambda)$-EA with $c \ge c_0 + \delta$.
2) The $(\mu+1)$-EA with $c \ge c_0 + \delta$.
3) The $(\mu+1)$-GA with $c \ge c_0 + \delta$ if $\mu = \mu(c)$ is sufficiently small.[14]
4) The $(1+(\lambda,\lambda))$-GA with $c\gamma \ge c_0 + \delta$.
5) The $(1+\lambda)$-fEA with any distribution satisfying (18) for $s \sim \mathcal{D}$.[13] In particular, this includes the following cases.
   a) The $(1+\lambda)$-fEA with $m_2/m_1 \ge 1+\delta$, if the probability to flip a single bit is sufficiently small compared to $s_0 := \min\{\sigma \in \mathbb{N} | m_{2,\le\sigma} \ge (1+\delta/2)m_1\}$, where $m_{2,\le\sigma} := \sum_{i=1}^{\sigma} \Pr[\mathcal{D}=i] i(i-1)$ is the truncated second falling moment.
   b) The $(1+\lambda)$-fEA with any power law distribution with exponent $\kappa \in (1,2)$, i.e., $\Pr[\mathcal{D} \ge \sigma] = \Omega(\sigma^{-\kappa})$.
   c) The $(1+\lambda)$-fEA with $\Pr[\mathcal{D}=1] \le (4/9)\cdot\Pr[\mathcal{D} \ge 3] - \delta$.
6) The $(\mu+1)$-fEA in all preceding cases for $(1+\lambda)$-fEA, if additionally $\Pr[\mathcal{D}=0] = \Omega(1)$.
7) The $(\mu+1)$-fGA in all preceding cases for $(1+\lambda)$-fEA, if the population size $\mu = \mu(\mathcal{D})$ is sufficiently small.[14]

Remark 1: The inequality $f(c,x) := cx - e^{-c(1-x)} - x/(1-x) \ge 0$ has a solution $x \in [0,1]$ if and only if $c \ge c_0$. To see this, it suffices to observe that the derivative with respect to c is $\partial f/\partial c(c,x) = x + (1-x)e^{-c(1-x)} > 0$. Hence, $f(c,x)$ is strictly increasing in c. The value of c_0, and the unique α_0 with $f(c_0,\alpha_0) = 0$ can numerically be computed as follows. Observe that $f(c_0,\alpha_0) = \partial f/\partial x(c_0,\alpha_0) = 0$. In particular, this implies $0 = c_0 f(c_0,\alpha_0) - \partial f/\partial x(c_0,\alpha_0) = (1 - c_0(1-\alpha_0) + c_0^2(1-\alpha_0^2)\alpha_0)/(1-\alpha_0)^2 =: \tilde{f}(c_0,\alpha_0)$. This is a quadratic equation in c_0 and has the two solutions $c_0 = g_\pm(\alpha_0) := (1 \pm \sqrt{1-4\alpha_0})/(2\alpha_0(1-\alpha_0))$ for $\alpha_0 \in (0, 1/4]$, and no solution otherwise. We can plug this term into the definition of $f(c,x)$, and obtain that α_0 is a root of $h_\pm(x) := f(g_\pm(x),x)$. The function h_- is strictly increasing in the interval $[0, 1/4]$ (the derivative can be checked to be positive in $(0, 1/4]$) from $h_-(0) = -1/e < 0$ to $h_-(1/4) = 1/3 - 1/e^2 > 0$, and thus it has a single zero in $[0, 1/4]$, which is $\alpha_0 = 0.237134\ldots$ The function h_+ is strictly decreasing from $h_+(0) = 1$ to $h_+(1/4) = 1/3 - 1/e^2 > 0$, and thus has no zero. Finally, c_0 can then be computed as the root of $\tilde{f}(c_0,\alpha_0) = 0$.

Remark 2: For the fEAs we remark that the interesting power-law regime $\kappa \in [2,3)$ is not excluded by the negative results in Theorem 6 if $\Pr[\mathcal{D}=1]$ is sufficiently large. In particular, a calculation with Mathematica shows that the Zipf distribution[15] with exponent $\kappa \ge 2$ satisfies (19) for all $\alpha \in (0,1)$. However, note that this holds only if the distribution is *exactly* the Zipf distribution; changing any probability even by a constant factor may lead to exponential runtimes. Moreover, it is rather questionable whether the Zipf distribution is efficient for *all* monotone functions, as $m_2/m_1 = \infty$ in this regime.

[13]Note that this is not a trivial consequence of Theorem 5, since (18) and (19) are conditions on the distribution for the best of λ offspring, while the condition here is on the distribution $\mathcal{D}$ for generating a single offspring.

[14]This statement follows trivially from the other results by setting $\mu = 1$, and it is listed only for completeness.

[15]That is, $\Pr[\mathcal{D}=k] = k^{-\kappa}/\zeta(\kappa)$, where ζ is the Riemann ζ function.

Proof of Theorem 6: All results will be applications of Theorem 5. We first outline the general strategy, for concreteness in the case of the $(1+\lambda)$-EA and $(1+\lambda)$-fEA. To apply Theorem 5 directly, we would need to analyze the distribution of the number of bit flips in the best offspring in each generation. Note crucially that this may be very different from the distribution $\mathcal{D}$ that creates a single offspring. However, the key feature of Theorem 5 is that it allows us to restrict our analysis to the case when the parent has at most ζn zero bits. Still the number of bit flips in the fittest offspring is not the same as $\mathcal{D}$, but we can use a neat trick. We "modify" the algorithm by choosing the winner offspring in a slightly different way. If none of the offspring flips a zero-bit, then we do not compare the *fittest* offspring with the parent x, but rather a *random* offspring. Otherwise, we proceed as usual with the fittest offspring. Note that this little thought experiment does not change the behavior of the algorithm, since in the former case *all* offspring are either identical to x, or have strictly worse fitness than x. So the algorithm just stays with the parent. However, if we call our weirdly selected winner offspring y, then suddenly the distribution $\mathcal{D}'$ of y is very similar to the distribution $\mathcal{D}$ of a random offspring, since most of the time we do not flip any zero-bits. We will be able to use the same trick for all the algorithms above, even for the population-based ones.

This construction would do the trick, except that it is not unbiased. However, note that if there is *exactly one* offspring $y^{(k)}$ which is fitter than x then the distribution of $y^{(j)}$ is unbiased conditioned on $f(y^{(j)}) > f(x^{(j)})$, i.e., the distribution of $y^{(j)}$ is the same as the (unbiased) distribution $\mathcal{D}$ of a single offspring, conditioned on this offspring being fitter. Therefore, we do have an unbiased distribution in all cases except for the case that there are at least two offspring which are fitter than x. We will attribute all these to the adversary. Thus, we need to show that the probability $\Pr[\text{at least two fitter offspring}|\text{at least one fitter offspring}] < \eta = \eta(\zeta', \alpha)$, and that we have a tail bound on the number of bit flips in this case. The tail bound follows as in the proof of Theorem 5 by observing that the probability to flip at least as many zero-bits as one-bits in A_i decreases exponentially in the number s of bit flips, if $d(A_i) \le 1/3$. In particular, the expected number of bit flips in a fitter offspring is $O(1)$. Hence, the probability to generate a better offspring is at most $O(\zeta)$, and so is the probability to generate a *second* fitter offspring. We can make this probability smaller than η by choosing ζ sufficiently small, since $\eta = \eta(\zeta', \alpha)$ is independent of ζ. This shows that the adversary is sufficiently limited.

Before we proceed to the individual algorithms, we first show in general why it suffices if $\mathcal{D}' \to \mathcal{D}$ weakly, which means the following. Let s and s' denote random variables from $\mathcal{D}$ and $\mathcal{D}'$, respectively, and let us denote $p_\sigma := \Pr[s = \sigma]$ and $p'_\sigma := \Pr[s' = \sigma]$. Then, we assume that for each $\sigma \in \mathbb{N}$ there is $\zeta_0 = \zeta_0(\sigma) > 0$ such that for all $0 < \zeta \le \zeta_0$ we have $p_\sigma = p'_\sigma \pm \xi$. We will show that under this assumption the value of Φ is approximately the same for s and s'. For convenience, we repeat the definition of Φ

$$\Phi = \Phi(x) = \frac{\mathbb{E}\left[s(s-1)(1-\alpha)^{s-1}\right]}{\mathbb{E}\left[s(1-\alpha)^{s-1}\right]} - \frac{\frac{1-\alpha}{\alpha}\Pr[s=1]}{\mathbb{E}\left[s(1-\alpha)^{s-1}\right]}.$$

We define Φ' analogously with s' instead of s. Consider the expectations in Φ. We can approximate each of them up to an error of ξ if we consider the contribution of the case $\sigma \le \sigma_0$ for the expectations. More precisely, for each $\xi > 0$ there is some constant $\sigma_0 \in \mathbb{N}$ such that $\mathbb{E}[s(1-\alpha)^{s-1}] = \sum_{\sigma=0}^{\sigma_0} p_\sigma \sigma (1-\alpha)^{\sigma-1} \pm \xi$ and $\mathbb{E}[s'(1-\alpha)^{s'-1}] = \sum_{\sigma=0}^{\sigma_0} p'_\sigma \sigma (1-\alpha)^{\sigma-1} \pm \xi$. Now, we use our assumption that for each $\sigma \in \{0, \dots, \sigma_0\}$ there is $\zeta_0 = \zeta_0(\sigma) > 0$ such that $p_\sigma = p'_\sigma \pm \xi$ whenever $0 < \zeta < \zeta_0$. Since we only want to achieve this for a constant number σ_0 of values, we can choose $\zeta_0 := \min\{\zeta_0(\sigma) | \sigma \in \{0, \dots, \sigma_0\}\} > 0$, and we obtain that $p_\sigma = p'_\sigma \pm \xi$ holds for all $\sigma \in \{0, \dots, \sigma_0\}$ simultaneously. Therefore,

$$\begin{aligned}
&\mathbb{E}\left[s(1-\alpha)^{s-1}\right] \\
&= \sum_{\sigma=0}^{\sigma_0} p_\sigma \sigma (1-\alpha)^{\sigma-1} \pm \xi \\
&= \sum_{\sigma=0}^{\sigma_0} \left(p'_\sigma \pm \xi\right) \sigma (1-\alpha)^{\sigma-1} \pm \xi \\
&= \sum_{\sigma=0}^{\sigma_0} p'_\sigma \sigma (1-\alpha)^{\sigma-1} \pm \xi \cdot \left(1 + \sum_{\sigma=0}^{\sigma_0} \sigma (1-\alpha)^{\sigma-1}\right) \\
&= \mathbb{E}\left[s'(1-\alpha)^{s'-1}\right] \pm \xi \cdot \left(2 + \sum_{\sigma=0}^{\infty} \sigma (1-\alpha)^{\sigma-1}\right).
\end{aligned}$$

Since the latter sum converges, we find that we can make the error term arbitrarily small by making $\xi > 0$ sufficiently small. The same argument applies to $\mathbb{E}[s(s-1)(1-\alpha)^{s-1}]$. Since, we can approximate each of these terms with arbitrary precision, and since all terms are finite and positive, we can make the error $\Phi - \Phi'$ arbitrarily small by choosing $\zeta > 0$ small enough. In particular, if $\Phi(x) < 1-\delta$ then $\Phi' < 1-\delta/2$ for ζ small enough, and we can apply Theorem 5.

Now, we turn more concretely to $(1+\lambda)$-EA and $(1+\lambda)$-fEA. In fact, the $(1+\lambda)$-EA is just a special case of the $(1+\lambda)$-fEA, where the number of bit flips is given by the binomial distribution $\text{BIN}(n, c/n)$, which converges to $\text{POI}(c)$ for $n \to \infty$. We first assume that $m_1 < \infty$. Recall that we consider the case that the parent x has at most ζn zero bits. Then, the probability that at least one of the λ offspring hits at least one zero-bit is at most $\Pr[\text{hit zero-bit}] \le \mathbb{E}[\text{zero-bit flips}] = \zeta \lambda m_1$. Hence, for every $\sigma \in \mathbb{N}$ we have $p'_\sigma = p_\sigma \pm \zeta \lambda m_1$. As outlined above, this implies that Φ' comes arbitrarily close to Φ if ζ is small enough. For the other case, $m_1 = \infty$, fix $\xi > 0$, and choose $\sigma_0 \in \mathbb{N}$ so large that $\Pr[s > \sigma_0] \le \xi/(2\lambda)$. Then by a union bound, the probability that at least one offspring flips more than σ_0 bits is at most $\xi/2$. On the other hand, if $s \le \sigma_0$ for all offspring then as in the previous case the probability to hit at least one zero-bit is at most $\zeta \lambda \sigma_0 \le \xi/2$, where the latter inequality is true for all $\zeta \le 2\lambda\sigma_0\xi$. With this choice, for every $\sigma \in \mathbb{N}$ we have $p_\sigma = p'_\sigma \pm \xi$, as required. Thus, we may evaluate Φ with respect to $\mathcal{D}$ instead of $\mathcal{D}'$.

Before we evaluate Φ, we remark that for the $(1+(\lambda,\lambda))$-GA, we can use almost the same argument with a slightly different construction of the winner offspring. For each of the λ offspring, we do λ crossover with the parent. If none of the λ^2 crossover offspring has a flipped zero-bit compared to x,

then we chose a random crossover offspring. Otherwise we choose the offspring as in the algorithm, i.e., we first pick the fittest mutation offspring z, and then pick the fittest crossover offspring of z. Since each of the λ^2 crossover offspring has in expectation $c\gamma$ flipped bits, the probability that there is some crossover offspring with a flipped zero bit is at most $\zeta\lambda^2 c\gamma$. Since this becomes arbitrarily small as ζ becomes small, the same argument applies, and we may evaluate Φ with respect to $\mathcal{D}$ instead of $\mathcal{D}'$.

It thus remains to evaluate Φ for various $\mathcal{D}$, and check that $\Phi \geq 1-\zeta$ or $\Phi \leq 1+\zeta$. For the $(1+\lambda)$-EA and the $(1+(\lambda,\lambda))$-GA we have a Poisson distribution $\text{Poi}(c)$. We use Mathematica to evaluate $\mathbb{E}[s(1-\alpha)^{s-1}] = ce^{-\alpha c}$, $\mathbb{E}[s(s-1)(1-\alpha)^{s-1}] = (1-\alpha)c^2e^{-\alpha c}$, and $\Pr[s=1] = ce^{-c}$, which leads to

$$\Phi = \Phi(\alpha, c) = \frac{1-\alpha}{\alpha}\Big(c\alpha - e^{-(1-\alpha)c}\Big).$$

We want to study whether there is $\alpha \in [0,1]$ with $\Phi(\alpha,c) \geq 1$. Rewriting this condition, this is the case if and only if there is an α such that the function $f(\alpha,c) := c\alpha - e^{-(1-x)\alpha} - (\alpha/[1-\alpha])$ takes non-negative values. For constant c, the function is negative for $\alpha = 0$ and $\alpha \to 1$. Therefore, the function takes non-negative values for this c if and only if $f(\cdot,c)$ has a zero, and c_0 is defined as the smallest value of c for which this happens. Moreover, the function is strictly increasing in c (see Remark 1), so any larger value of c will admit some value of α for which the function is strictly positive. This proves the statements for the $(1+\lambda)$-EA.

For the $(1+\lambda)$-fEA, we first consider the case $(m_2/m_1) \leq 1-\delta$. In this case we may bound the second term of Φ by 0, hence

$$\Phi \leq \frac{\mathbb{E}\big[s(s-1)(1-\alpha)^{s-1}\big]}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]} \overset{(*)}{\leq} \frac{\mathbb{E}[s(s-1)]}{\mathbb{E}[s]} = \frac{m_2}{m_1} \leq 1-\delta$$

where (*) follows from Chebyshev's sum inequality since the factor $(s-1)$ is increasing and $(1-\alpha)^{s-1}$ is decreasing in s. This settles the cases in which the $(1+\lambda)$-fEA is successful.

For the second part of the theorem, we have already shown that a distribution satisfying (19) for some $0<\alpha<1$ needs exponential time. It remains to show that (19) is satisfied in the special cases listed in the theorem. Assume first that $m_2/m_1 \geq 1+\delta$, and that $\Pr[s=1] \leq 1/(Cs_0)$, where $C>0$ is a sufficiently large constant that we choose later. Here, s_0 is as in the theorem, i.e., $m_{2,\leq s_0} = \sum_{\sigma=1}^{s_0} p_\sigma \sigma(\sigma-1) \geq (1+\delta/2)m_1$. Since the condition $m_2/m_1 > 1+\delta$ stays true if we make δ smaller, we may assume $\delta \leq 1/10$. Choose $\alpha := 1/(C's_0)$, where $C' := 16/\delta$. Then, $\alpha \leq 1/2$, which implies $1-\alpha \geq e^{-2\alpha}$. Hence, for all $\sigma \in [s_0]$ we have $(1-\alpha)^\sigma \geq (1-\alpha)^{s_0} \geq e^{-2\alpha s_0} = e^{-2/C'} \geq 1-2/C'$. Therefore,

$$\begin{aligned}\mathbb{E}\Big[s(s-1)(1-\alpha)^{s-1}\Big] &\geq \sum_{\sigma=1}^{s_0} p_\sigma s(s-1)\Big(1-\frac{2}{C'}\Big)\\ &\geq \Big(1+\frac{\delta}{2}\Big)m_1\Big(1-\frac{2}{C'}\Big)\\ &\geq \Big(1+\frac{\delta}{4}\Big)m_1.\end{aligned}$$

Moreover, if we choose $C \geq 8C'/(\delta m_1)$, then we may bound

$$\frac{1-\alpha}{\alpha}\Pr[s=1] \leq C's_0 \cdot \frac{1}{Cs_0} = \frac{C'}{C} \leq \frac{\delta}{8}m_1.$$

Plugging this into Φ, we get that

$$\begin{aligned}\Phi &= \frac{\mathbb{E}\big[s(s-1)(1-\alpha)^{s-1}\big] - \frac{1-\alpha}{\alpha}\Pr[s=1]}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]}\\ &\geq \frac{\big(1+\frac{\delta}{8}\big)m_1}{\mathbb{E}[s]} = 1+\frac{\delta}{8}\end{aligned}$$

as required.

The second special case for the $(1+\lambda)$-fEA is that $\mathcal{D}$ is a power law distribution with exponent $\kappa \in (1,2)$, i.e., $p_\sigma = \Theta(\sigma^{-\kappa})$. This case is similar as the previous case, since we have for all $s_0 \in \mathbb{N}$

$$m_{2,\leq s_0} = \sum_{\sigma=1}^{s_0} p_\sigma \sigma(\sigma-1) = \sum_{\sigma=1}^{s_0} \Theta\Big(\sigma^{2-\kappa}\Big) = \Omega\Big(s_0^{3-\kappa}\Big) = \omega(s_0)$$

where the Landau notation in this case is with respect to $s_0 \to \infty$ instead of $n \to \infty$. For $\alpha := 1/s_0$ we have $1-\alpha \geq e^{-2\alpha}$ and thus $(1-\alpha)^{s_0} \geq e^{-2}$. Hence, if s_0 is a sufficiently large constant

$$\begin{aligned}\mathbb{E}\Big[s(s-1)(1-\alpha)^{s-1}\Big] &\geq e^{-2}m_{2,\leq s_0} \geq \frac{2}{\delta}s_0 = \frac{2}{\delta}\cdot\frac{1}{\alpha}\\ &\geq \frac{2}{\delta}\cdot\frac{1-\alpha}{\alpha}\Pr[s=1].\end{aligned}$$

Therefore,

$$\Phi \geq \frac{\big(1-\frac{\delta}{2}\big)\mathbb{E}\big[s(s-1)(1-\alpha)^{s-1}\big]}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]} \geq \frac{\big(1-\frac{\delta}{2}\big)m_2}{m_1} \geq 1+\frac{\delta}{4}$$

where the last step holds for all $\delta \leq 1/2$.

The third special case for the $(1+\lambda)$-fEA is that $p_1 \leq (4/9)\cdot p_3 - \delta$. In this case, choose $\alpha = 1/3$ and observe that $3p_3(1-\alpha)^2 \geq p_1/\alpha + 3\delta$. Using this

$$\begin{aligned}&\sum_{\sigma=1}^{3} p_\sigma\sigma(\sigma-1)(1-\alpha)^{\sigma-1} - \frac{1-\alpha}{\alpha}p_1\\ &\quad\geq 2p_2(1-\alpha) + 3p_3(1-\alpha)^2 + \frac{p_1}{\alpha} + 3\delta - \frac{1-\alpha}{\alpha}p_1\\ &\quad= \sum_{\sigma=1}^{3} p_\sigma\sigma(1-\alpha)^{\sigma-1} + 3\delta.\end{aligned}$$

Hence,

$$\begin{aligned}\Phi &\geq \frac{\sum_{\sigma=1}^{3} p_\sigma\sigma(1-\alpha)^{\sigma-1} + 3\delta + \sum_{\sigma=4}^{\infty} p_\sigma\sigma(\sigma-1)(1-\alpha)^{\sigma-1}}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]}\\ &\geq \frac{\mathbb{E}\big[s(1-\alpha)^{s-1}\big] + 3\delta}{\mathbb{E}\big[s(1-\alpha)^{s-1}\big]} = 1+\Omega(1).\end{aligned}$$

This settles the last case of the $(1+\lambda)$-fEA.

So far we have analyzed all cases with $\mu = 1$, so let us turn to $\mu > 1$. We first give a general argument for the case that all individuals in the population have at most ζn zero bits, for some sufficiently small constant $\zeta > 0$ which may depend on the constants in the theorem (e.g., on μ). In the following, we will use the Landau notation only to hide factors that are

independent of ζ. For example, $A = O(B)$ means that B/A is bounded by some constant which is independent of ζ (but which may depend on the constants in the theorem).

Assume that $x^{(1)}$ is a search point of maximal fitness in the current population, and let y be the offspring in the current generation. We observe that $\Pr[y = x^{(1)}] = \Omega(1)$: for the GAs there is a constant probability that y is generated by crossing $x^{(1)}$ with itself, since $\mu = O(1)$. For the EAs there is a constant probability to make a mutation without bit flips. (For the fEAs and fGAs this is an explicit condition.) Moreover, since $x^{(1)}$ has maximal fitness in the population, with constant probability both $x^{(1)}$ and y survive the selection step. (They may be eliminated if the whole population has the same fitness.) By the same argument, there is a small, but $\Omega(1)$ probability that $x^{(1)}$ duplicates in μ successive rounds, and all its offspring survive all $\mu - 1$ selection steps. Hence, with probability $\Omega(1)$ the population degenerates to μ copies of the same individuals. We say in this case that the search point and the round are *consolidated*. Since this happens in each batch of μ rounds with constant probability, the expected time until some search point is consolidated is $O(1)$.

We first consider the $(\mu + 1)$-EA and $(\mu + 1)$-fEA. To apply Theorem 5, we will reinterpret the $(\mu+1)$ algorithms as follows. Assume that at some point in time t_1 there is a consolidated search point with at most $\zeta n/2$ zero bits. We call this search point $x^{(1)}$. Then, we define recursively t_i to be the minimal $t > t_{i-1}$ such that there is a consolidated search point at time t. We define $x^{(i)}$ to be the consolidated search point at time t_i. In this way, the sequence of $x^{(i)}$ fits the description of an algorithm in Theorem 5, although the process of going from $x^{(i-1)}$ to $x^{(i)}$ is rather complex. To complete the description, we still need to define the offspring x' that appears in the algorithm description in Theorem 5. We define it as the first offspring that is created from x. If x' or x are consolidated, then this fits the definition of the algorithm.[16] Otherwise we blame it to the adversary. Thus we need to show that the adversary is limited as required by Theorem 5. Note that the distribution of x' is just the distribution $\mathcal{D}$ of the mutation operator. Thus, the same results as for the $(1+\lambda)$-EA and $(1+\lambda)$-fEA immediately carry over if we can show that the adversary is sufficiently limited.

To estimate the effect of the adversary, assume that the current consolidated search point x has at most ζn zero bits, and consider the first mutant x' with $f(x') > f(x)$. Afterwards, the population consists of $\mu - 1$ copies of x and one copy of x'. In each subsequent round, there are four (nonexclusive) possibilities.

1) Another copy of x is created. This happens with probability $\Omega(1)$.
2) Another copy of x' is created. This happens with probability $\Omega(1)$.
3) A mutation $\neq x, x'$ is created with no flipped zero bit.
4) A mutation is created with at least one flipped zero bits. This happens with probability $O(\zeta)$.

Note that until 4) happens, all search points in the population are either equal to x or x', or are strictly dominated by x or x'. In particular, all search points in the population are either copies of x', or have a strictly worse fitness than x'. Therefore, x' will be consolidated after an expected $O(1)$ number of steps, unless case 4) occurs before that. Thus, the probability that x' is consolidated before case 4) occurs is $1 - O(\zeta)$. This means that the adversary may only act with probability $O(\zeta)$, which is sufficiently small if ζ is small.

It remains to estimate the number τ of bits in which x differs from the next consolidated search point y. Note that in any sequence of $\mu = O(1)$ rounds, we have probability $\Omega(1)$ that the population is consolidated, so the probability that we see at least k rounds before consolidation drops exponentially in k. This implies that $\mathbb{E}[\tau] = O(1)$. Note that it would already imply exponentially falling tail bounds on τ for the $(\mu + 1)$-EA, but for the general case of the $(\mu + 1)$-fEA we need to use a similar argument as for the $(1+\lambda)$-fEA, as follows.

The next consolidated search point y must satisfy $\mathrm{HT}(y) \geq \mathrm{HT}(x)$, since otherwise it could not supersede x in the population. Let $i := \ell(x) + 1$ be the index of the current hot topic, and let s be the number of bit flips to create x'. Then, $\mathrm{HT}(x') \geq \mathrm{HT}(x)$ can only happen if either the level increases, or $d(A_i)$ does not increase. Thus, $\Pr[\mathrm{HT}(x') \geq \mathrm{HT}(x) | s = \sigma] = e^{-\Omega(\sigma)}$, since the number of one-bits in A_i *increases* in expectation by $\alpha\sigma(1 - 2\zeta) = \Omega(\sigma)$, and likewise for the number of one-bits in B_{i+1}. Hence, $\Pr[s \geq \sigma] = e^{-\Omega(\sigma)}$. Similarly, if x'' is the next offspring (either from x or from x'), then the probability that x'' survives selection in this round decreases exponentially in the number s'' of bit flips, since $\Pr[\mathrm{HT}(x'') \geq \mathrm{HT}(x)] = e^{-\Omega(s'')}$. Repeating this argument, for any fixed number of rounds the total number of bit flips in these rounds has an exponential tail bound. Since the number of rounds before consolidation has also an exponential tail bound, this proves the exponential tail bound on τ. This concludes the proof for the $(\mu + 1)$-EA and $(\mu + 1)$-fEA.

Note for later use that the same tail bound argument also applies for the $(\mu + 1)$-GA and $(\mu + 1)$-fGA since crossover can only change bits that have been touched since the last consolidated round. So the total number of bits that are touched between two consolidated rounds does not increase.

For the $(\mu+1)$-GA and $(\mu+1)$-fGA, note that the exponential runtime statements for small μ follow trivially from the $(1+1)$-EA and $(1+1)$-fEA, since they agree with $(\mu+1)$-GA and $(\mu + 1)$-fGA if $\mu = 1$. So let us consider the upper runtime bounds for large μ. The situation is similar to the one for $(\mu + 1)$-EA and $(\mu + 1)$-fEA, but with the crucial difference that the errors made in the creation of x' may be repaired by crossovers between x and x'. Other than before, we will apply part 3) of Theorem 5.

Assume as before that x is a consolidated search point. Note that crossovers cannot create new search points, so assume that an offspring $x' \neq x$ with $\mathrm{HT}(x') > \mathrm{HT}(x)$ is created from x by a mutation. Let S_{01} and S_{10} be the sets of bits that were flipped from zero to one and from one to zero, respectively,

[16] In fact, if $f(x) = f(x')$ then this does not quite fit the description of the algorithm, since we might consolidate x while the elitist algorithm would always choose x'. However, the function HT is symmetric with respect to any search points which have the same fitness, i.e., for any two such search points there is an automorphism of $\{0, 1\}^d$ which interchanges the search points, but which leaves HT invariant. Thus it does not matter which of the two search points we choose.

and let $s_{01} = |S_{01}|$, $s_{10} = |S_{10}|$, and $s = s_{01} + s_{10}$. Note that $s_{01} > 0$. As before, we have $\Pr[s \geq \sigma] = e^{-\Omega(\sigma)}$. Let σ_0 be a constant such that $\Pr[s \geq \sigma_0] \leq \eta/4$, where η is the constant from Theorem 5. Note that by making ζ small enough, we can also bound the probability that an additional zero-bit is flipped before the next consolidated round by $\eta/4$. So assume that $s \leq \sigma_0$ and that no additional zero-bits are flipped until consolidation.

Let k be the number of search points in the population that are not copies of x. In rounds where the parents of mutation or crossover are picked among the copies of x, the population does not change. Otherwise, i.e., when at least one parent is not x, there is a probability of at least $1/k$ that it is a copy of y. With probability at least $1/2$ the operation in this round is a crossover (crossover has a larger probability than mutation since two parents are picked). Moreover, if $k \leq \mu/2$, then with probability at least $1/2$ it is a crossover with x. Therefore, the expected number of crossover children between x and x' is at least $\sum_{k=1}^{\mu/2} 1/(4k) \geq (1/4)\ln(\mu/2)$. Note that this becomes arbitrarily large if μ is large. In particular, for sufficiently large μ, with probability at least $1 - \eta/4$ there will be at least C_2 crossovers between x and x' in expectation, for any desired constant $C_2 > 0$.

Now observe that since $s \leq \sigma_0$, every crossover copy between x and x' has probability at least $2^{-\sigma_0} = \Omega(1)$ to retain the bits in S_{01} from x' and to pick all bits in S_{10} from x. In particular, if μ is large enough, then with probability $1 - \eta/4$ this happens at least once. If it happens, then the offspring y dominates x, x', and any crossover of x, and x. Moreover, since we assume that no zero-bits are flipped by mutations, it also dominates any mutation offspring of any search point in the population. Therefore, the population must consolidate with y. In this case we say that x' was *fully repaired*. Note that whenever an offspring x' of a consolidated search point x with at most ζn zero bits is created with $\mathrm{HT}(x') > \mathrm{HT}(x)$, then it has probability at least $1 - \eta$ to be fully repaired.

We are now ready to explain how we apply Theorem 5 (c). As before, let $x^{(1)}$ be the first consolidated search point. Then, we define recursively t_i to be the minimal $t > t_{i-1}$ such that there is a consolidated search point at time t, and such that at least one mutation happens in rounds $t_{i-1}, \ldots, t$. The latter condition simply means that we ignore rounds in which a consolidated search point performs a crossover with itself. We define $x^{(i)}$ to be the consolidated search point at time t_i, and we define $x'^{(i)}$ to be the first mutation offspring after time t_i. If x has more than ζn zero bits then we define the winner offspring $y^{(i)}$ to be $x'^{(i)}$, which gives an unbiased distribution. If x has at most ζn zero-bits, then we define $y^{(i)} := x^{(i)}$ if $\mathrm{HT}(x'^{(i)}) \leq \mathrm{HT}(x^{(i)})$, and we define $y^{(i)}$ to be the fully repaired $x'^{(i)}$ if $\mathrm{HT}(x'^{(i)}) \leq \mathrm{HT}(x^{(i)})$. In the latter case, we have shown that indeed $x^{(i+1)} = y^{(i)}$ with probability at least $1 - \eta$, so we may blame any other outcome to the adversary. The power of the adversary is limited in the same way as for the $(\mu+1)$-EA and $(\mu+1)$-fEA, so we may indeed apply Theorem 5 (c). This concludes the proof. ∎

Remark 3: We remark that the constructions for $(1+\lambda)$-EA and $(\mu+1)$-EA can be combined, and that the same arguments carry over and give the same runtimes as in Theorem 6 for the $(\mu+\lambda)$-EA, and likewise for the $(\mu+\lambda)$-fEA. However, due to the considerable complexity of the individual arguments, we refrain from proving this statement.

VI. Conclusion

We have studied a large set of algorithms, and we have shown that in all cases without crossover, there is a dichotomy with respect to a parameter (c, $c\gamma$, or Φ, where the latter one is related to m_2/m_1) for optimizing the monotone function family HotTopic. If the parameter is small, then the algorithms need time $O(n \log n)$; if the parameter is large then they need exponential time on some instances. In the cases $(1+\lambda)$-EA, $(1+1)$-fEA $(1+(\lambda,\lambda))$-GA, and for good start points also $(1+\lambda)$-fEA, if the parameter is small, then we could show that the algorithms are actually fast on *all* monotone functions. However, there are many open problems left, and we conclude this paper by a selection of those.

1) We have analyzed the algorithms theoretically for the case $n \to \infty$. Experiments would be interesting to understand for what values of n the effects become observable. For example, do larger values of λ and μ *delay* the detrimental effect of HotTopic, so that it is only visible for larger n?[17]
2) In some cases our runtime bounds for small parameter values hold only for HotTopic, but the general status of monotone functions remains unclear ($(\mu+1)$-EA, $(\mu+1)$-fEA). So does a small mutation parameter guarantee a small runtime on *all* monotone functions?
3) We could show that genetic algorithms are superior to EAs on the HotTopic functions. However, is the same true in general for monotone functions? Is it true that the $(\mu+1)$-GA and $(\mu+1)$-fGA are fast for all monotone functions if μ is large enough?
4) It seems important to understand more precisely how large μ should be in GAs to cope with larger mutation parameters. For example, for the $(\mu+1)$-GA with mutation parameter c, how large does μ need to be so that it is still fast on all HotTopic instances?
5) What are the smallest mutation parameters for which there are hard monotone functions? E.g., for the $(1+1)$-EA, what is the smallest $\tilde{c}$ such that the runtime is polynomial for all $c < \tilde{c}$? We know that $\tilde{c} \leq c_0$, and it was recently shown that $\tilde{c} > 1$ [21], but neither upper nor lower bound seem to be tight. Moreover, for $c < \tilde{c}$, is the runtime always $O(n \log n)$?
6) Our proofs for population sizes $\mu > 1$ rely on the fact that in all considered algorithms diversity is lost close to the optimum. Do the results stay the same if diversity is actively maintained, for example by duplication avoidance [27], [31] or by genotypical or phenotypical niching [30]?
7) How is the performance of algorithms that change the mutation strength dynamically (parameter control), e.g., with the 1/5-th rule? In the introduction we have given an intuition why this might be risky, so what rules can optimize HotTopic efficiently?

[17]The supplementary online material contains a few experimental results, which show that the dichotomy can also be clearly observed in data. However, the set of experiments is much too small to give a fine-grained picture.

8) While HOTTOPIC is defined in a discrete setting, the underlying intuition is related to continuous optimization. Is there a continuous analogue of HOTTOPIC, and what is the performance of optimization algorithms like the CMA-ES or particle swarm optimization?

ACKNOWLEDGMENT

Part of the work was inspired by discussions at the Dagstuhl meeting 19171 on Theory of Randomized Optimization Heuristics. In particular, the author thank B. Doerr for proposing to consider the $(1 + \lambda)$-EA on monotone functions, which started this line of research. The study of monotone functions was fostered by the COST Action CA15140 "Improving Applicability of Nature-Inspired Optimisation by Joining Theory and Practice," and the author has proposed to include monotone functions into the set of benchmarks developed by working group 3.

REFERENCES

[1] F. Chicano, A. M. Sutton, L. D. Whitley, and E. Alba, "Fitness probability distribution of bit-flip mutation," *Evol. Comput.*, vol. 23, no. 2, pp. 217–248, 2015.
[2] D. Corus and P. S. Oliveto, "Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms," *IEEE Trans. Evol. Comput.*, vol. 22, no. 5, pp. 720–732, Oct. 2018.
[3] B. Doerr, "Optimal parameter settings for the $(1+(\lambda, \lambda))$ genetic algorithm," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2016, pp. 1107–1114.
[4] B. Doerr and C. Doerr, "Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2015, pp. 1335–1342.
[5] B. Doerr and C. Doerr, "A tight runtime analysis of the $(1+(\lambda, \lambda))$ genetic algorithm on OneMax," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2015, pp. 1423–1430.
[6] B. Doerr and C. Doerr, "Optimal static and self-adjusting parameter choices for the $(1+(\lambda, \lambda))$ genetic algorithm," *Algorithmica*, vol. 80, no. 5, pp. 1658–1709, 2018.
[7] B. Doerr, C. Doerr, and F. Ebel, "Lessons from the black-box: Fast crossover-based genetic algorithms," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2013, pp. 781–788.
[8] B. Doerr, C. Doerr, and F. Ebel, "From black-box complexity to designing new genetic algorithms," *Theor. Comput. Sci.*, vol. 567, pp. 87–104, Feb. 2015.
[9] B. Doerr, T. Jansen, D. Sudholt, C. Winzen, and C. Zarges, "Optimizing monotone functions can be difficult," in *Parallel Problem Solving From Nature, PPSN*. Heidelberg, Germany: Springer, 2010.
[10] B. Doerr, T. Jansen, D. Sudholt, C. Winzen, and C. Zarges, "Mutation rate matters even when optimizing monotonic functions," *Evol. Comput.*, vol. 21, no. 1, pp. 1–27, Mar. 2013.
[11] B. Doerr, D. Johannsen, and C. Winzen, "Multiplicative drift analysis," *Algorithmica*, vol. 64, no. 4, pp. 673–697, 2012.
[12] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2017, pp. 777–784.
[13] C. Doerr and J. Lengler, "Introducing elitist black-box models: When does elitist behavior weaken the performance of evolutionary algorithms?" *Evol. Comput.*, vol. 25, no. 4, pp. 587–606, 2017.
[14] T. Friedrich, A. Göbel, F. Quinzan, and M. Wagner, "Heavy-tailed mutation operators in single-objective combinatorial optimization," in *Proc. PPSN*, Coimbra, Portugal, 2018, pp. 134–145.
[15] T. Friedrich, F. Quinzan, and M. Wagner, "Escaping large deceptive basins of attraction with heavy-tailed mutation operators," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2018, pp. 293–300.
[16] G. H. Hardy, J. E. Littlewood, and G. Pólya, *Inequalities*. Cambridge, U.K.: Cambridge Univ. Press, 1988.
[17] J. Jägersküpper, "Combining Markov-chain analysis and drift analysis," *Algorithmica*, vol. 59, no. 3, pp. 409–424, 2011.
[18] T. Jansen, "On the brittleness of evolutionary algorithms," in *Foundations of Genetic Algorithms, FOGA*. Heidelberg, Germany: Springer, 2007.
[19] P. K. Lehre and C. Witt, "Black-box search by unbiased variation," *Algorithmica*, vol. 64, no. 4, pp. 623–642, 2012.
[20] J. Lengler, "A general dichotomy of evolutionary algorithms on monotone functions," in *Proc. PPSN*, Coimbra, Portugal, 2018, pp. 3–15.
[21] J. Lengler, A. Martinsson, and A. Steger, "When does hillclimbing fail on monotone functions: An entropy compression argument," in *Proc. Anal. Algorithms Combinatorics (ANALCO)*, 2019, pp. 94–102.
[22] J. Lengler and A. Steger, "Drift analysis and evolutionary algorithms revisited," *Combinatorics Probab. Comput.*, vol. 27, no. 4, pp. 643–666, 2018.
[23] V. Mironovich and M. Buzdalov, "Evaluation of heavy-tailed mutation operator on maximum flow test generation problem," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2017, pp. 1423–1426.
[24] P. S. Oliveto and C. Witt, "Simplified drift analysis for proving lower bounds in evolutionary computation," *Algorithmica*, vol. 59, no. 3, pp. 369–386, 2011.
[25] P. S. Oliveto and C. Witt, "Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation," *arXiv preprint arXiv:1211.7184*, 2012.
[26] P. S. Oliveto and C. Witt, "Improved time complexity analysis of the simple genetic algorithm," *Theor. Comput. Sci.*, vol. 605, pp. 21–41, Nov. 2015.
[27] E. C. Pinto and C. Doerr, "A simple proof for the usefulness of crossover in black-box optimization," in *Proc. PPSN*, Coimbra, Portugal, 2018, pp. 29–41.
[28] Y. Pushak and H. H. Hoos, "Algorithm configuration landscapes: More benign than expected?" in *Proc. PPSN*, Coimbra, Portugal, 2018, pp. 271–283.
[29] J. E. Rowe and D. Sudholt, "The choice of the offspring population size in the $(1, \lambda)$ evolutionary algorithm," *Theor. Comput. Sci.*, vol. 545, pp. 20–38, Aug. 2014.
[30] O. M. Shir, "Niching in evolutionary algorithms," in *Handbook of Natural Computing*. Heidelberg, Germany: Springer, 2012, pp. 1035–1069.
[31] D. Sudholt, "How crossover speeds up building block assembly in genetic algorithms," *Evol. Comput.*, vol. 25, no. 2, pp. 237–274, 2017.
[32] C. Witt, "Tight bounds on the optimization time of a randomized search heuristic on linear functions," *Combinatorics Probab. Comput.*, vol. 22, no. 2, pp. 294–318, 2013.

Johannes Lengler received the Diplom degree in mathematics from Saarland University, Saarbrücken, Germany, in 2002, the M.Sc. degree from Warwick University, Coventry, U.K., in 2005, and the Ph.D. degree in algebraic number theory from Saarland University in 2010.

He has a broad research spectrum, reaching from pure mathematics to neuroscience. He joined ETH Zürich, Zürich, Switzerland, as a Post-Doctoral Fellow, where he is currently an Senior Scientist, and simultaneously entered the fields of computer science and of neuroscience. His current research interests include nature-inspired optimization algorithms, random graph models for large real-world networks, and stochastic processes in graphs and networks in computer science; and neuroplasticity, memory, and rapid learning in neuroscience. His other publications are concerned with efficient communication protocols, the power of decision-makers with limited choices, computational geometry, and quantum computing.