A General Dichotomy of Evolutionary Algorithms on Monotone Functions

Johannes Lengler(✉)

Department of Computer Science, ETH Zürich, Zürich, Switzerland
johannes.lengler@inf.ethz.ch

Abstract. It is known that the $(1 + 1)$-EA with mutation rate c/n optimises every monotone function efficiently if $c < 1$, and needs exponential time on some monotone functions (HotTopic functions) if $c > c_0 = 2.13692...$ We study the same question for a large variety of algorithms, particularly for $(1 + \lambda)$-EA, $(\mu + 1)$-EA, $(\mu + 1)$-GA, their fast counterparts like fast $(1 + 1)$-EA, and for $(1 + (\lambda, \lambda))$-GA. We prove 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.

1 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 standard bit mutation, i.e., all bits are flipped independently. On the one hand, if the mutation strength is too low, then 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 already 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 $1/n$ [16]. This rate minimises the expected runtime, i.e., the expected number of function evaluations before the optimum is hit. Any deviation from this mutation rate to either direction decreases performance.

Extended Abstract. All proofs and further details are available on arxiv [12].

A. Auger et al. (Eds.): PPSN 2018, LNCS 11102, pp. 3–15, 2018.
https://doi.org/10.1007/978-3-319-99259-4_1

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 $f(x) > f(y)$. In particular, every monotone function has a unique global optimum at $(1 \dots 1)$. Moreover, every such function is efficiently optimised by *random local search* (RLS), which is the (1+1) algorithm that flips in each round exactly one random bit. From any starting point, RLS finds the optimum after at most n improving steps, and by a coupon collector argument it will optimise any monotone function in time $O(n \log n)$. Thus, monotone functions might be regarded as trivial to optimise, and we might expect every standard EA to solve them efficiently.

However, this is not so. Doerr et al. showed [7,8] that even the $(1 + 1)$-EA, which flips each bit independently with 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, but for large mutation rate, $c > 16$, there are monotone functions for which the (1+1)-EA needs exponential time. Lengler and Steger [13] gave a simpler construction of such "hard" monotone functions, which we call HotTopic (they didn't provide a name), and which yield exponential runtime for $c > c_0 := 2.13692...$ The basic idea of this construction is that at every point in time there is some subset of bits which form a"hot topic", i.e., the algorithm considers them much more important than the other bits. 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 analyse whether they can overcome the detrimental effect of the HotTopic functions for larger mutation rates. In particular, we consider (for constant μ, λ) the classical $(1 + \lambda)$-EA, $(\mu + 1)$-EA, and $(\mu + 1)$-GA, the $(1 + (\lambda, \lambda))$-GA by Doerr et al. [6], and the recently proposed fast $(1 + \lambda)$-EA, fast $(\mu + 1)$-EA, and fast $(\mu + 1)$-GA [9], which we abbreviate 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!

Results. In this section we collect our results for the different algorithms. Note that, unless explicitly otherwise stated, we always assume that the parameters μ, λ, c, γ of the algorithms are constant.

Classical EAs. For the classical evolutionary algorithm $(1 + \lambda)$-EA, we show a dichotomy: if the mutation parameter c is sufficiently small, then the algorithms optimise 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

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

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 $c > c_0$, 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 optimise all HotTopic functions in time $O(n \log n)$, while for $c > c_0$ it is exponentially slow on some HotTopic instances. The same result on HotTopic holds for the $(\mu+1)$-EA. 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 can not exclude that the situation might get even *worse* for larger μ, as there may still be other monotone functions which are hard for the $(\mu+1)$-EA with $c < 1$.

$(\mu+1)$-GA. It has been observed before that some algorithms may be sped up by crossover. In particular, Sudholt [15] 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. Thus it remains an intriguing open question whether the $(\mu+1)$-GA with large μ is fast on *every* monotone function.

$(1+(\lambda,\lambda))$-GA. This algorithm creates λ offspring, and uses the best of them to perform λ biased crossovers with the parent, see Sect. 2. The best crossover offspring is then compared with the parent. This algorithm has been derived by Doerr et al. [5,6] from a theoretical understanding of so-called *black-box complexity*, and has been intensively studied thereafter [1–4]. Most remarkably, it gives an asymptotic improvement on the runtime of the most intensively studied test function OneMax, on which it has 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 non-constant $\lambda = \lambda(n)$. The $(1+(\lambda,\lambda))$-GA is arguably the only known natural unbiased evolutionary algorithm that can optimise 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 [4] it is suggested to choose c, γ in such a way that $c\gamma = 1$. Note that this makes sense, because $c\gamma$ is (neglecting 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 $c\gamma > c_0$ 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 optimise HotTopic.

Notably, the runtime benefits on OneMax carry over, at least to the HotTopic function. Since the benefits on OneMax in previous work have been achieved for non-constant parameter choices, we relax our assumption on constant parameters for the $(1 + (\lambda, \lambda))$-GA. In particular, we show that for the optimal static parameter and adaptive parameter settings in [9], the algorithm achieves the same asymptotic runtime on HotTopic as on OneMax, in particular runtime $O(n)$ in the adaptive setup.

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 this mutation is always selected for crossovers. In the most relevant regime, where the expected number of flipped zero-bits in *any* mutation is small (say, at most one), the probability of being selected increases by a factor of $\Theta(\lambda)$ (from $1/\lambda$ to $\Theta(1)$) if a zero-bit is flipped. For monotone functions the probability to be selected does 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.

Fast (1+1)-EA, **Fast (1+λ)-EA**, **Fast (μ+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. [9], and they have immediately attracted considerable attention (e.g, [14]). 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 mutation is generated from the parent by flipping exactly s bits. 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: of small mutation rate and of 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.[2] 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 $m_2/m_1 > c_0$ and additionally $p_1 := \Pr[\mathcal{D} = 1]$ is sufficiently small, then the runtime on some HotTopic instances is exponential. As for the other functions, 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 (Eq. (5) on page 9), it

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

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 [9,14]. Also, no distribution with $p_1 < \frac{4}{9}\Pr[\mathcal{D} = 3]$ is efficient on HotTopic. In general, our findings contrast the results in [9], where larger tails (smaller κ) lead to faster runtimes.

As before, without crossover larger values of λ and μ do not seem to have any effect. 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 it is still possible that larger values of λ, μ make things even worse.

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 optimises 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. 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. For example, simulations suggest for the $(1+1)$-EA that $\varepsilon \approx 0.15$ for $c = 4$. 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 initialised with random search points with εn incorrect bits, then still the runtime is exponential.

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 evolutionary algorithms. All EAs are highly vulnerable even to a very moderate increase of the mutation rate. 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.

Intuition on HotTopic. We conclude the introduction with an intuition why the HotTopic functions are hard to optimise 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 [13] 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 [13] 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 behaviour that is very good at optimising this particular aspect – but all other aspects will deteriorate a little because they are out of focus. Thus if the hot topic is rather 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 just a pessimistic allegory on scientific progress, but it also describes evolutionary algorithms with large mutation rates. They rank the currently preferred direction above everything else, and 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 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 small, and improvements often occur in aggressive steps in which many bits are flipped. Then the same step may cause many 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, so an adaptive algorithm might react by increasing the mutation parameter. This indeed increases the probability to find a better search point in the hot topic direction (though not the probability to make *any* improvement), and may thus lead fatally to a large mutation parameter.

2 Preliminaries and Definitions

Notation. Throughout the 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 $f(x) > f(y)$.[3] We will consider maximisation algorithms, and we will mostly focus on the *runtime* of an algorithm, i.e., the number of function evaluations before the algorithm evaluates the global maximum of f for the first time. We say that an EA or GA is *elitist* [10] if the selection operator greedily chooses the fittest individuals to form the next generation. We call an EA or GA *unbiased* [11] if the mutation and crossover 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 $[n] := \{1, \dots, n\}$. We will use n for the dimension of the search space, μ for the population size, λ for the offspring population size, 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

[3] Note that this property might more correctly be called *strictly monotone*, but in this paper we will stick with the shorter, slightly less precise term *monotone*. In all other cases we use the standard terminology, e.g. the term *increasing sequence* has the same meaning as *non-decreasing sequence*.

and second falling moment $\mathbb{E}[s \mid s \sim \mathcal{D}]$ and $\mathbb{E}[s(s-1) \mid s \sim \mathcal{D}]$, respectively. Unless otherwise stated, we will assume that $\mu, \lambda, c, \gamma = \Theta(1)$ and $m_1 = \Omega(1)$.

Algorithms. Most algorithms that we consider fall into the class of $(\mu + \lambda)$ evolutionary algorithms, $(\mu + \lambda)$-*EAs*, or $(\mu + \lambda)$ genetic algorithms, $(\mu + \lambda)$-*GAs*. They can be described as follows. 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. We will always assume that $\mu, \lambda, c = \Theta(1)$.

An exception to the above scheme is the $(1 + (\lambda, \lambda))$-GA [5]. Here the population consists of a single search point x. Then in each round, we pick $s \sim \textsc{Bin}(n, c/n)$, 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.

Hard Monotone Functions: HotTopic. In this section we give the construction of hard monotone functions by Lengler and Steger [13], following closely their exposition. The functions come with four parameters $\alpha, \beta, \rho, \varepsilon$, and they are given by a randomised construction. We call the corresponding function $\textsc{HotTopic}_{\alpha,\beta,\rho,\varepsilon} = \mathrm{HT}_{\alpha,\beta,\rho,\varepsilon} = \mathrm{HT}$. 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.

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 $\ell(x) := \max\{\ell' \in [e^{\rho n}] : |\{j \in B_{\ell'} : x_j = 0\}| \le \varepsilon\beta n\}$, where we set $\ell(x) = 0$, if no such ℓ' exists). Then we define $f : \{0,1\}^n \to \mathbb{R}$ as follows:

$$\mathrm{HT}(x) := \ell(x) \cdot n^2 + \sum\nolimits_{i \in A_{\ell(x)+1}} x_i \cdot n + \sum\nolimits_{i \notin A_{\ell(x)+1}} x_i, \tag{2}$$

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 [13] that whp[4] the $(1+1)$-EA with $c > c_0$ needs exponential time to find the optimum. One easily checks that this function is monotone: the (monotone) term $\ell(x)n^2$ dominates the rest, and for constant values of ℓ the remaining terms just give a linear function.

3 Results

We first give a generic result for strong dichotomies, i.e., we specify circumstances under which an algorithm optimises every monotone function in time $O(n \log n)$.

Theorem 1 (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 $x \in \{0,1\}^n$,*

$$\mathbb{E}[s_{10} \mid \textit{parent} = x \textit{ and } s_{01} > 0] \leq 1 - \delta, \tag{3}$$

and

$$\Pr[s_{01} > 0 \mid \textit{parent} = x] = \Omega(\tfrac{1}{n}(n - \textsc{OneMax}(x))). \tag{4}$$

Then whp the runtime of $\mathcal{A}$ on any (strictly) monotone functions is $O(n \log n)$.

The $(1+\lambda)$-EA, the $(1+1)$-fEA, and the $(1+(\lambda,\lambda))$-GA all fit the generic description in Theorem 1, modulo Condition (3). For the $(1+(\lambda,\lambda))$-GA, 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. The crucial step is in all cases to show that these settings satisfy (3). For the $(1+(\lambda,\lambda))$-GA with $c\gamma < 1$ and non-constant parameters we cannot apply Theorem 1 directly. However, the proof is similar, and the conditional expectation in (3) is still the crucial object to study.

Theorem 2. *Let $\delta > 0$. The following algorithms need whp $O(n \log n)$ generations on any (strictly) monotone function.*

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

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

Moreover, if the $(1 + (\lambda, \lambda))$*-GA with* $c\gamma < 1 - \delta$ *uses the optimal static or adaptive parameter choice from [4][5], then whp 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 [4,6].

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 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 an artefact of our proof.

Theorem 3. *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, 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* $O(n \log n)$ *whp.*

Next we analyse the behaviour 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 4 (HotTopic, Generic Runtime). *Let* $0 < \alpha < 1$*. Consider an elitist, unbiased optimisation 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*

$$\Phi(x) := \frac{\mathbb{E}[s(s-1)(1-\alpha)^{s-1}]}{\mathbb{E}[s(1-\alpha)^{s-1}]} - \frac{((1-\alpha)/\alpha) \cdot \Pr[s=1]}{\mathbb{E}[s(1-\alpha)^{s-1}]}. \tag{5}$$

(a) If there are constants $\zeta, \zeta' > 0$ *such that*

$$\Phi(x) \geq 1 + \zeta' \tag{6}$$

holds for all $x \in \{0,1\}^n$ *with at most* ζn *zero-bits, then whp* $\mathcal{A}$ *has exponential runtime on* $\mathrm{HotTopic}_{\alpha,\beta,\rho,\varepsilon}$ *with parameters* β, ρ, ε *as in* (1)*.*

[5] In fact, the suggested parameter choice in [4,6] satisfies $c\gamma = 1$ instead of $c\gamma < 1$. However, the runtime analysis in [6] only changes by constant factors if γ is decreased by a constant factor. Thus Theorem 2 applies to the parameter choices from [4,6], except that γ is decreased by a constant factor.

(b) If there are constants $\zeta, \zeta' > 0$ such that

$$\Phi(x) \leq 1 - \zeta' \tag{7}$$

holds for all $x \in \{0,1\}^n$ with at most ζn zero-bits, and if moreover $\Pr[s = 1] \geq \zeta$ and $\mathbb{E}[s(s-1)] \leq 1/\zeta$ for all parents x, then whp $\mathcal{A}$ has runtime $O(n \log n)$ on $\textsc{HotTopic}_{\alpha,\beta,\rho,\varepsilon}$ with parameters β, ρ, ε as in (1).

(c) The statements in (a) and (b) remain true for algorithms that are only unbiased conditioned on an improving step, if in (b) we have $\Pr[\text{improving step}] \geq \zeta \cdot d([n], x)$ as well. Moreover, (b) 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 (7) by the condition $\mathbb{E}[s \mid \mathrm{HT}(y) > \mathrm{HT}(x)] \leq 2 - \zeta$.

Finally, there is a constant $\eta = \eta(\zeta', \alpha) > 0$ independent of ζ such that (a), (b), and (c) 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 she may change up to τ bits in the current search point. For (a) we additionally require $\Pr[\tau \geq \tau'] = e^{-\Omega(\tau')}$, while for (b) and (c) we only require $\Pr[\tau \geq n^{1-\eta}] = o(1/(n \log n))$.

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

It turns out that Theorem 4 suffices to classify the behaviour 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 4 explicitly requires population size one. However, for small ε the populations typically collapse to μ identical copies of the same search point, and the other cases can be attributed to the adversary. In this way Theorem 4 implies the following theorem.

Theorem 5 (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 $c_0 = 2.13692..$ 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 $\alpha \in (0,1)$, whp each of the following algorithms optimises the function $\textsc{HotTopic}_{\alpha,\beta,\rho,\varepsilon}$ with parameters β, ρ, ε as in (1) in time $O(n \log n)$.*

- *The $(1+\lambda)$-EA with $c \leq c_0 - \delta$.*
- *The $(\mu+1)$-EA with $c \leq c_0 - \delta$.*
- *The $(\mu+1)$-GA with arbitrary $c = \Theta(1)$ if $\mu = \mu(c)$ is sufficiently large.*
- *The $(1 + (\lambda, \lambda))$-GA with $c\gamma \leq c_0 - \delta$.*
- *The $(1+\lambda)$-fEA with $m_2/m_1 \leq 1 - \delta$; more generally, the $(1+\lambda)$-fEA with any distribution that satisfies (7) for $s \sim \mathcal{D}$, as well as $\Pr[\mathcal{D} = 1] = \Omega(1)$.*[6]

[6] Note that this is not a trivial consequence of Theorem 4, since (6), (7) 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.

- *The* $(\mu+1)$*-fEA in the preceding case, if additionally* $\Pr[\mathcal{D}=0]=\Omega(1)$.
- *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 $\alpha_0 = 0.237134..$*, whp each of the following algorithms needs exponential time to optimise the function* $\textsc{HotTopic}_{\alpha_0,\beta,\rho,\varepsilon}$ *with parameters* β,ρ,ε *as in* (1).

- *The* $(1+\lambda)$*-EA with* $c \geq c_0+\delta$.
- *The* $(\mu+1)$*-EA with* $c \geq c_0+\delta$.
- *The* $(\mu+1)$*-GA with* $c \geq c_0+\delta$ *if* $\mu=\mu(c)$ *is sufficiently small.*[7]
- *The* $(1+(\lambda,\lambda))$*-GA with* $c\gamma \geq c_0+\delta$.
- *The* $(1+\lambda)$*-fEA with any distribution satisfying* (6) *for* $s\sim\mathcal{D}$.[6] *In particular, this includes the following cases.*
  - *The* $(1+\lambda)$*-fEA with* $m_2/m_1 \geq 1+\delta$*, if* $\Pr[\mathcal{D}=1] \geq C/s_0$ *for a sufficiently large constant* $C>0$*, where* $s_0 := \min\{\sigma\in\mathbb{N} \mid m_{2,\leq\sigma} \geq (1+\delta/2)m_1\}$*, with* $m_{2,\leq\sigma} := \sum_{i=1}^{\sigma}\Pr[\mathcal{D}=i]i(i-1)$.
  - *The* $(1+\lambda)$*-fEA with any power law distribution with exponent* $\kappa\in(1,2)$*, i.e.* $\Pr[\mathcal{D}\geq\sigma]=\Omega(\sigma^{-\kappa})$.
  - *The* $(1+\lambda)$*-fEA with* $\Pr[\mathcal{D}=1] \leq \frac{4}{9}\cdot\Pr[\mathcal{D}\geq 3]-\delta$.
- *The* $(\mu+1)$*-fEA in all preceding cases for* $(1+\lambda)$*-fEA, if* $\Pr[\mathcal{D}=0]=\Omega(1)$.
- *The* $(\mu+1)$*-fGA in all preceding cases for* $(1+\lambda)$*-fEA if* $\mu=\mu(\mathcal{D})$ *is sufficiently small.*[7]

Remark 1. For the fEAs we remark that the interesting regime $\kappa\in[2,3)$ is not excluded by the negative results in Theorem 5, if $\Pr[\mathcal{D}]$ is sufficiently large. In particular, a calculation with Mathematica$^{\text{TM}}$ shows that the Zipf distribution[8] with exponent $\kappa\geq 2$ satisfies (7) 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.

4 Conclusions

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 optimising 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 of $(1+\lambda)$-EA, $(1+1)$-fEA $(1+(\lambda,\lambda))$-GA, and for good start points also of $(1+\lambda)$-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 the paper by a selection of those.

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

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

- We have analysed the algorithms theoretically for the case $n \to \infty$. Experiments are sorely needed to understand the effects for small finite n.
- 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?
- We could show that genetic algorithms are superior to evolutionary algorithms on the HotTopic functions. However, is the same true in general for monotone functions? Is it true that the $(\mu+1)$-GA and the $(\mu+1)$-fGA are fast for all monotone functions if μ is large enough?
- 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 m need to be so that it is still fast on all HotTopic instances?
- By now a classical question is: are there monotone functions which are hard for the parameter range $[1, c_0)$? Most intriguingly: are there hard monotone instances for the $(1+1)$-EA for every $c > 1$? For $c = 1$ it is known that the runtime is polynomial, but is it always $O(n \log n)$?
- Our proofs for population sizes $\mu > 1$ rely on the fact that in all considered algorithms diversity tends to be lost close to the optimum. Do the results stay the same if diversity is actively maintained, for example by duplication avoidance or by genotypical or phenotypical niching?
- How is the performance of algorithms that change the mutation strength dynamically, e.g., with the 1/5-th rule? The introduction gives an intuition why this might be bad, but intuition has failed before on monotone functions.
- While HotTopic is defined in a discrete setting, the underlying intuition is related to continuous optimisation. Is there a continuous analogue of HotTopic, and what is the performance of optimisation algorithms like the CMA-ES or particle swarm optimisation?

References

1. Doerr, B.: Optimal parameter settings for the $(1 + \lambda, \lambda)$ genetic algorithm. In: GECCO (2016)
2. Doerr, B., Doerr, C.: Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings. In: GECCO (2015)
3. Doerr, B., Doerr, C.: A tight runtime analysis of the $(1 + (\lambda, \lambda))$ genetic algorithm on OneMax. In: GECCO (2015)
4. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the $(1+(\lambda, \lambda))$ genetic algorithm. Algorithmica **80**, 1–52 (2017)
5. Doerr, B., Doerr, C., Ebel, F.: Lessons from the black-box: fast crossover-based genetic algorithms. In: GECCO (2013)
6. Doerr, B., Doerr, C., Ebel, F.: From black-box complexity to designing new genetic algorithms. Theor. Comput. Sci. **567**, 87–104 (2015)
7. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Optimizing monotone functions can be difficult. In: Schaefer, R., Cotta, C., Kołodziej, J., Rudolph, G. (eds.) PPSN 2010. LNCS, vol. 6238, pp. 42–51. Springer, Heidelberg (2010). https://doi.org/10.1007/978-3-642-15844-5_5

8. Doerr, B., Jansen, T., Sudholt, D., Winzen, C., Zarges, C.: Mutation rate matters even when optimizing monotonic functions. Evol. Comput. **21**(1), 1–27 (2013)
9. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: GECCO (2017)
10. Doerr, C., Lengler, J.: Introducing elitist black-box models: when does elitist behavior weaken the performance of evolutionary algorithms? Evol. Comput. **25**(4), 587–606 (2017)
11. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**, 623–642 (2012)
12. Lengler, J.: A general dichotomy of evolutionary algorithms on monotone functions. arXiv e-prints (2018)
13. Lengler, J., Steger, A.: Drift analysis and evolutionary algorithms revisited. arXiv e-prints (2016)
14. Mironovich, V., Buzdalov, M.: Evaluation of heavy-tailed mutation operator on maximum flow test generation problem. In: GECCO (2017)
15. Sudholt, D.: How crossover speeds up building block assembly in genetic algorithms. Evol. Comput. **25**(2), 237–274 (2017)
16. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Comb. Probab. Comput. **22**(2), 294–318 (2013)