Significance-based Estimation-of-Distribution Algorithms

Benjamin Doerr
École Polytechnique
Laboratoire d'Informatique (LIX)
Palaiseau, France

Martin S. Krejca
Hasso Plattner Institute
Potsdam, Germany

ABSTRACT

Estimation-of-distribution algorithms (EDAs) are randomized search heuristics that maintain a stochastic model of the solution space. This model is updated from iteration to iteration based on the quality of the solutions sampled according to the model. As previous works show, this short-term perspective can lead to erratic updates of the model, in particular, to bit-frequencies approaching a random boundary value. This can lead to significant performance losses.

In order to overcome this problem, we propose a new EDA that takes into account a longer history of samples and updates its model only with respect to information which it classifies as statistically significant. We prove that this *significance-based compact genetic algorithm* (sig-cGA) optimizes the common benchmark functions OneMax and LeadingOnes both in $O(n \log n)$ time, a result shown for no other EDA or evolutionary algorithm so far. For the recently proposed scGA – an EDA that tries to prevent erratic model updates by imposing a bias to the uniformly distributed model – we prove that it optimizes OneMax only in a time exponential in the hypothetical population size $1/\rho$.

CCS CONCEPTS

• **Theory of computation → Theory and algorithms for application domains**; *Theory of randomized search heuristics*;

KEYWORDS

Estimation-of-distribution algorithm; theory; run time analysis

ACM Reference Format:
Benjamin Doerr and Martin S. Krejca. 2018. Significance-based Estimation-of-Distribution Algorithms. In *GECCO '18: Genetic and Evolutionary Computation Conference, July 15–19, 2018, Kyoto, Japan.* ACM, New York, NY, USA, 8 pages. https://doi.org/10.1145/3205455.3205553

1 INTRODUCTION

Estimation-of-distribution algorithms (EDAs; [22]) are a special class of *evolutionary algorithms* (EAs). They optimize a function by evolving a stochastic model of the solution space. In an iterative fashion, an EDA uses its stochastic model to generate samples and then updates it with respect to observations made from these samples. An algorithm-specific parameter determines how drastic the changes to the model in each iteration are. In order for an EDA to succeed in optimization, it is important that the stochastic model is changed over time such that better solutions are sampled more frequently. However, due to the randomness in sampling, the model should not be changed too drastically in a single iteration in order to prevent wrong updates from having a long-lasting impact.

The theory of EDAs has recently gained momentum [3, 11, 12, 15, 16, 23, 25, 26] and is mainly concerned with the aforementioned trade-off of the convergence speed of an EDA to a near-optimal model while making sure that the model does not prematurely converge to suboptimal models. This trade-off is very visible in the results of Sudholt and Witt [23] and Krejca and Witt [15], who prove lower bounds of the expected run times of three common EDAs on the benchmark function OneMax. In simple words, these bounds show that if the parameter for updating the model is too large, the model converges too quickly and very likely to a wrong model; in consequence, it then takes a long time to find the optimum (usually by first reverting to a better fitting model). On the other hand, if the parameter is too small, then the model does converge to the correct model, but it does so slowly.

The problem of how to choose the parameter has also been discussed by Friedrich et al. [11]. They consider a class of EDAs that all current theoretical results fall into: *n-Bernoulli-λ-EDAs* optimizing functions over bit strings of length n. The stochastic model of such EDAs uses one variable per bit of a bit string, resulting in a vector of probabilities τ of length n called the *frequency vector*. In each iteration, a bit string x is sampled bit-wise independently and independent of any other sample such that bit x_i is 1 with probability *(frequency)* τ_i and 0 otherwise. Thus, the stochastic model used by such EDAs is a Poisson-binomial distribution. Friedrich et al. [11] consider two different properties of such EDAs: *balanced* and *stable*. Intuitively, in expectation, a *balanced* EDA does not change a frequency τ_i if the fitness function has no bias toward 0s or 1s at that respective position i. A *stable* EDA keeps a frequency, in such a scenario, close to 1/2. Friedrich et al. [11] then prove that an EDA cannot be both balanced and stable. This means that the frequencies will always move toward 0 or 1, even if there is no bias from the objective function *(fitness function)*. They also prove that all commonly theoretically analyzed EDAs are balanced.

The results of Friedrich et al. [11], Sudholt and Witt [23], and Krejca and Witt [15] draw the following picture: for a balanced EDA, there exists some inherent noise in the update. Thus, if the parameter responsible for the update of the stochastic model is large and the speed of convergence high, the algorithm only uses a few samples before it converges. During this time, the noise introduced by the balanced-property may not be overcome, resulting in the

ACM ISBN 978-1-4503-5618-3/18/07…$15.00
https://doi.org/10.1145/3205455.3205553

[1]The results shown for PBIL are the results of UMDA, since the latter is a special case of the former. Wu et al. [26] also analyze PBIL but with worse results.

Table 1: Expected run times (number of fitness evaluations) of various algorithms until they first find an optimum for the two functions OneMax and LeadingOnes (eq. (1)). For optimal parameter settings, many algorithms have a run time of $\Theta(n \log n)$ for OneMax and of $\Theta(n^2)$ for LeadingOnes. We note that the $(1 + (\lambda, \lambda))$ GA has an $o(n \log n)$ run time on OneMax (and even linear run time with a dynamic parameter choice), but we do not see why it should have a performance better than quadratic on LeadingOnes. Further, we strongly believe that the CSA has an exponential run time on OneMax.

Algorithm	OneMax	constraints	LeadingOnes	constraints
$(1+1)$ EA	$\Theta(n \log n)$ [9]	none	$\Theta(n^2)$ [9]	none
$(\mu+1)$ EA	$\Theta(\mu n + n \log n)$ [24]	$\mu = O\big(\mathrm{poly}(n)\big)$	$\Theta(\mu n \log n + n^2)$ [24]	$\mu = O\big(\mathrm{poly}(n)\big)$
$(1+\lambda)$ EA	$\Theta\left(n \log n + \frac{\lambda n \log\log\lambda}{\log\lambda}\right)$ [6, 14]	$\lambda = O(n^{1-\varepsilon})$	$\Theta(n^2 + \lambda n)$ [14]	$\lambda = O\big(\mathrm{poly}(n)\big)$
$\big(1+(\lambda,\lambda)\big)$ GA	$\Theta\left(\max\left\{\frac{n \log n}{\lambda}, \frac{n\lambda \log\log\lambda}{\log\lambda}\right\}\right)$ [5]	$p = \frac{\lambda}{n}, c = \frac{1}{\lambda}$	unknown	–
CSA	unknown	–	$O(n \log n)$ [18]	$\mu \geq 8\ln\big((4n+6)n\big)$, restarts
UMDA/PBIL[1]	$\Omega(\lambda\sqrt{n} + n \log n)$ [15] $O(\lambda n)$ [16, 25]	$\mu = \Theta(\lambda)$ $\mu = \Omega(\log n) \cap O(\sqrt{n}), \lambda = \Omega(\mu)$ or $\mu = \Omega(\sqrt{n}\log n), \mu = \Theta(\lambda)$ or $\mu = \Omega(\log n) \cap o(n), \mu = \Theta(\lambda)$	$O(n\lambda \log\lambda + n^2)$ [3]	$\lambda = \Omega(\log n), \mu = \Theta(\lambda)$
cGA/2-MMAS$_{\mathrm{ib}}$	$\Omega\left(\frac{\sqrt{n}}{\rho} + n \log n\right)$ [23] $O\left(\frac{\sqrt{n}}{\rho}\right)$ [23]	$\frac{1}{\rho} = O\big(\mathrm{poly}(n)\big)$ $\frac{1}{\rho} = \Omega(\sqrt{n}\log n) \cap O\big(\mathrm{poly}(n)\big)$	unknown	–
1-ANT	$\Theta(n \log n)$ [19]	$\rho = \Theta(1)$	$O(n^2 \cdot 2^{5/(n\rho)})$ [7] $2^{\Omega(\min\{n, 1/(n\rho)\})}$ [7]	none none
scGA (Alg. 2)	$\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ [Thm. 4.1]	$1/\rho = \Omega(\log n), a = \Theta(\rho)$, $d = \Theta(1), c > 0$	$O(n \log n)$ [11]	$1/\rho = \Theta(\log n), a = O(\rho), d = \Theta(1)$
sig-cGA (Alg. 1)	$O(n \log n)$ [Thm. 3.6]	$\varepsilon > 12$	$O(n \log n)$ [Thm. 3.4]	$\varepsilon > 12$

stochastic model converging to an incorrect one, as the algorithms are not stable. Hence, the parameter should be smaller in order to guarantee convergence to the correct model, resulting in a slower optimization time.

The core problem in this dilemma lies in what information the EDAs use in order to perform an update: the current samples and the current frequency vector – no time-dependent information. Thus, the algorithms are forced to make an on-the-spot decision with respect to how to update their frequency vector. This entails that they will most likely make a change to their model although this change may be harmful.[2] Thus, Friedrich et al. [11] propose an EDA (called *scGA*) that is stable (but not balanced) in order to converge quicker to a correct frequency vector by introducing an artificial bias into the update process that should counteract the bias of a balanced EDA. However, this approach fails on the standard benchmark function OneMax, as we prove in this paper (Thm. 4.1).

We propose a new approach that tries to eliminate the aforementioned problems by introducing a new EDA that is aware of its history of samples: the *significance-based compact genetic algorithm* (sig-cGA). For each position, it has access to a part of the history of bits sampled so far. If it detects that either statistically significantly more 1s than 0s or vice versa were sampled, it changes the corresponding frequency, otherwise not. Thus, the sig-cGA only performs an update when it has proof that it makes sense. This sets it apart from the other EDAs analyzed so far. We prove that the sig-cGA is able to optimize OneMax and LeadingOnes in $O(n \log n)$ in expectation and with high probability (Thm. 3.6 and 3.4), which has not been proven before for any other EDA or classical EA (for further details, see Table 1). Further, we prove that the scGA, which is known to have an expected run time of $O(n \log n)$ on LeadingOnes [11] too, is not able to optimize OneMax in that time.

[2]Note that *balanced* only means that a frequency does not change *in expectation*.

Our paper is structured as follows: Section 2 establishes some notation and the setting we consider. In Section 3, we introduce and discuss our new algorithm sig-cGA. We also go into detail how the extra information of the sig-cGA can be efficiently implemented such that the additional overhead is small. Further, we prove that the sig-cGA optimizes LeadingOnes and OneMax in $O(n \log n)$ in expectation and with high probability (Thm. 3.4 and 3.6, respectively). In Section 4, we shortly discuss the scGA and prove that it optimizes OneMax in $\Omega(2^{\Theta(1/\rho)})$ (Thm. 4.1), where ρ is an algorithm-specific parameter. We conclude our paper in Section 5.

2 PRELIMINARIES

In this work, we consider the maximization of pseudo-Boolean functions $f\colon \{0,1\}^n \to \mathbb{R}$, where n is a positive integer (fixed for the remainder of this work). We call f a *fitness function*, an element $x \in \{0,1\}^n$ an *individual*, and, for an $i \in [n] := [1,n] \cap \mathbb{N}$, we denote the ith bit of x by x_i. When talking about *run time*, we always mean the number of fitness function evaluations of an algorithm until an optimum is sampled for the first time.

In our analysis, we regard the two classic benchmark functions OneMax and LeadingOnes defined by

$$\textsc{OneMax}(x) = \sum_{i \in [n]} x_i \text{ and } \textsc{LeadingOnes}(x) = \sum_{i \in [n]} \prod_{j \in [i]} x_j \,. \quad (1)$$

Algorithm 1: The sig-cGA with parameter ε and significance function sig (eq. (2)) optimizing f

```
1  $t \leftarrow 0$;
2  for $i \in [n]$ do $\tau_i^{(t)} \leftarrow \frac{1}{2}$ and $H_i \leftarrow \emptyset$ ;
3  repeat
4      $x, y \leftarrow$ offspring sampled with respect to $\tau^{(t)}$;
5      $x \leftarrow$ winner of $x$ and $y$ with respect to $f$;
6      for $i \in [n]$ do
7          $H_i \leftarrow H_i \circ x_i$;
8          if $\mathrm{sig}(\tau_i^{(t)}, H_i) = \mathrm{up}$ then $\tau_i^{(t+1)} \leftarrow 1 - 1/n$;
9          else if $\mathrm{sig}(\tau_i^{(t)}, H_i) = \mathrm{down}$ then $\tau_i^{(t+1)} \leftarrow 1/n$;
10         else $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)}$;
11         if $\tau_i^{(t+1)} \neq \tau_i^{(t)}$ then $H_i \leftarrow \emptyset$;
12     $t \leftarrow t + 1$;
13 until termination criterion met;
```

In other words, OneMax returns the number of 1s of an individual, whereas LeadingOnes returns the longest sequence of consecutive 1s of an individual, starting from the left. Note that the all-1s bit string is the unique global optimum for both functions.

We state in Table 1 the asymptotic run times of a few algorithms on these benchmark functions. We note that (i) the black-box complexity of OneMax is $\Theta(n/\log n)$, see [2, 10], and (ii) the black-box complexity of LeadingOnes is $\Theta(n \log\log n)$, see [1], however, all black-box algorithms witnessing these run times are highly artificial. Consequently, $\Theta(n \log n)$ appears to be the best run time to aim for for these two benchmark problems.

Since random bit strings with independently sampled entries occur frequently in this work, we shall regularly use the following well-known variance-based additive Chernoff bounds (see, e.g., the respective Chernoff bound in [4]).

Theorem 2.1 (Variance-based Additive Chernoff Bounds). *Let $X_1, \ldots, X_n$ be independent random variables such that, for all $i \in [n]$, $E[X_i] - 1 \leq X_i \leq E[X_i] + 1$. Further, let $X = \sum_{i=1}^n X_i$ and $\sigma^2 = \sum_{i=1}^n \mathrm{Var}[X_i] = \mathrm{Var}[X]$. Then, for all $\lambda \geq 0$, abbreviating $m = \min\{\lambda^2/\sigma^2, \lambda\}$,*

$$\Pr[X \geq E[X] + \lambda] \leq e^{-\frac{1}{3}m} \text{ and } \Pr[X \leq E[X] - \lambda] \leq e^{-\frac{1}{3}m} .$$

Further, we say that an event A occurs with high probability if there is a $c = \Omega(1)$ such that $\Pr[A] \geq (1 - n^{-c})$.

Last, we use the $\circ$ operator to denote string concatenation. For a bit string $H \in \{0,1\}^*$, let $|H|$ denote its length, $\|H\|_0$ its number of 0s, $\|H\|_1$ its number of 1s, and, for a $k \in [|x|]$, let $H[k]$ denote the *last* k bits in H. In addition to that, $\emptyset$ denotes the empty string.

3 THE SIGNIFICANCE-BASED COMPACT GENETIC ALGORITHM

Before presenting our algorithm sig-cGA in detail in Section 3.1, we provide more information about the *compact genetic algorithm* (cGA [13]), which the sig-cGA as well as the scGA are based on.

The cGA is an *estimation-of-distribution algorithm* (EDA [22]). That is, it optimizes a fitness function by evolving a stochastic model of the search space $\{0,1\}^n$. The cGA assumes independence of the bits in the search space, which makes it a univariate EDA. As such, it keeps a vector of probabilities $(\tau_i)_{i\in[n]}$, often called *frequency vector*. In each iteration, two individuals *(offspring)* are sampled in the following way with respect to the frequency vector: for an individual $x \in \{0,1\}^n$, we have $x_i = 1$ with probability τ_i, and $x_i = 0$ with probability $1 - \tau_i$, independently of any τ_j with $j \neq i$. Thus, the stochastic model of the cGA is a Poisson-binomial distribution.

After sampling, the frequency vector is updated with respect to a fitness-based ranking of the offspring. The process of choosing how the offspring are ranked is called *selection*. Let x and y denote both offspring of the cGA during an iteration. Given a fitness function f, we rank x above y if $f(x) > f(y)$ (as we maximize), and we rank y above x if $f(y) > f(x)$. If $f(x) = f(y)$, we rank them randomly. The higher-ranked individual is called the *winner*, the other individual the *loser*. Assume that x is the winner. The cGA changes a frequency τ_i then with respect to the difference $x_i - y_i$ by a value of ρ (where $1/\rho$ is usually referred to as population size). Hence, no update is performed if the bit values are identical, and the frequency is moved to the bit value of the winner. In order to prevent a frequency τ_i getting stuck at 0 or 1,[3] the cGA usually caps its frequency to the range $[1/n, 1 - 1/n]$, as is common practice. This way, a frequency can get close to 0 or 1, but it is always possible to sample 0s and 1s.

Consider a position i and any two individuals x and y that are identical except for position i. Assume that $x_i > y_i$. If the probability that x is the winner of the selection is higher than y being the winner, we speak of a *bias in selection* (for 1s) at position i. Analogously, we speak of a bias for 0s if the probability that y wins is higher than the probability that x wins. Usually, a fitness function introduces a bias into the selection and thus into the update.

3.1 Detailed Description of the sig-cGA

Our new algorithm – the *significance-based compact genetic algorithm* (sig-cGA; Alg. 1) – also samples two offspring each iteration. However, in contrast to the cGA, it keeps a history of bit values for each position and only performs an update when a statistical significance within a history occurs. This approach far better aligns with the intuitive reasoning that an update should only be performed if there is valid evidence for a different frequency being better suited for sampling good individuals.

In more detail, for each bit position $i \in [n]$, the sig-cGA keeps a history $H_i \in \{0,1\}^*$ of all the bits sampled by the winner of each iteration since the last time τ_i changed – the last bit denoting the latest entry. Observe that if there is no bias in selection at position i, the bits sampled by τ_i follow a binomial distribution with a success probability of τ_i and $|H_i|$ tries. We call this our *hypothesis*. Now, if we happen to find a sequence (starting from the latest entry) in H_i that significantly deviates from the hypothesis, we update τ_i with respect to the bit value that occurred significantly, and we reset the history. We only use the following three frequency values:

- $1/2$: starting value;
- $1/n$: significance for 0s was detected;
- $1 - 1/n$: significance for 1s was detected.

We formalize *significance* by defining the threshold for all $\varepsilon, \mu \in \mathbb{R}^+$, where μ is the expected value of our hypothesis and ε is an algorithm-specific parameter. $s(\varepsilon, \mu) = \varepsilon \max\{\sqrt{\mu \ln n}, \ln n\}$.

[3] A frequency τ_i at one of these two values results in the offspring only having the same bit value at position i. Thus, the cGA would not change τ_i anymore.

We say, for an $\varepsilon \in \mathbb{R}^+$, that a binomially distributed random variable X deviates significantly from a hypothesis $Y \sim \mathrm{Bin}(k, \tau)$, where $k \in \mathbb{N}^+$ and $\tau \in [0, 1]$, if there exists a $c = \Omega(1)$ such that $\Pr\big[|X - E[Y]| \le s(\varepsilon, E[Y])\big] \le n^{-c}$.

We now state our significance function sig: $\{\frac{1}{n}, \frac{1}{2}, 1 - \frac{1}{n}\} \times \{0, 1\}^* \to \{\mathrm{up}, \mathrm{stay}, \mathrm{down}\}$, which scans a history for a significance. However, it does not scan the entire history but multiple subsequences of a history (always starting from the latest entry). This is done in order to quickly notice a change from an insignificant history to a significant one. Further, we only check in steps of powers of 2, as this is faster than checking each subsequence and we can be off from any length of a subsequence by a constant factor of at most 2. More formally, for all $H \in \{0, 1\}^*$, we define, with ε being a parameter of the sig-cGA. Recall that $H[k]$ denotes the last k bits of H.

$$\mathrm{sig}\Big(\frac{1}{2}, H\Big) = \begin{cases} \mathrm{up} & \text{if } \exists m \in \mathbb{N}\colon \|H[2^m]\|_1 \ge \frac{2^m}{2} + s\big(\varepsilon, \frac{2^m}{2}\big), \\ \mathrm{down} & \text{if } \exists m \in \mathbb{N}\colon \|H[2^m]\|_0 \ge \frac{2^m}{2} + s\big(\varepsilon, \frac{2^m}{2}\big), \\ \mathrm{stay} & \text{else.} \end{cases}$$

$$\mathrm{sig}\Big(1 - \frac{1}{n}, H\Big) = \begin{cases} \mathrm{down} & \text{if } \exists m \in \mathbb{N}\colon \|H[2^m]\|_0 \ge \frac{2^m}{n} + s\big(\varepsilon, \frac{2^m}{n}\big), \\ \mathrm{stay} & \text{else.} \end{cases}$$

$$\mathrm{sig}\Big(\frac{1}{n}, H\Big) = \begin{cases} \mathrm{up} & \text{if } \exists m \in \mathbb{N}\colon \|H[2^m]\|_1 \ge \frac{2^m}{n} + s\big(\varepsilon, \frac{2^m}{n}\big), \\ \mathrm{stay} & \text{else.} \end{cases} \tag{2}$$

We stop at the first (minimum) length 2^m that yields a significance. Thus, we check a history H in each iteration at most $\log_2 |H|$ times.

We now prove that the probability of detecting a significance at a position when there is no bias in selection (i.e., a *false significance*) is small. We use this lemma in our proofs in order to argue that no false significances are detected with high probability.

Lemma 3.1. *For the sig-cGA (Alg. 1), let $\varepsilon \ge 1$. Consider a position $i \in [n]$ of the sig-cGA and an iteration such that the distribution X of 1s of H_i follows a binomial distribution with k trials and success probability τ_i, i.e., there is no bias in selection at position i. Then the probability that τ_i changes in this iteration is at most $n^{-\varepsilon/3} \log_2 k$.*

Proof. In order for τ_i to change, the number of 0s or 1s in X needs to deviate significantly from the hypothesis, which follows the same distribution as X by assumption. We are going to use Theorem 2.1 in order to show that, in such a scenario, X will deviate significantly from its expected value only with a probability of at most $n^{-\varepsilon/3} \log_2 k$ for any number of trials at most k.

Let $\tau'_i = \min\{\tau_i, 1 - \tau_i\}$. Note that, in order for τ_i to change, a significance of values sampled with probability τ'_i needs to be sampled. That is, for $\tau_i = 1/2$, either a significant amount of 1s or 0s needs to occur; for $\tau_i = 1 - 1/n$, a significant amount of 0s needs to occur; and, for $\tau_i = 1/n$, a significant amount of 1s needs to occur. Further, let X' denote the number of values we are looking for a significance within $k' \le k$ trials. That is, if $\tau_i = 1/2$, X' is either the number of 1s or 0s; if $\tau_i = 1 - 1/n$, X' is the number of 0s; and if $\tau_i = 1/n$, X' is the number of 1s.

Given the definition of τ'_i, we see that $E[X'] = k'\tau'_i$ and $\mathrm{Var}[X'] = k'\tau_i(1 - \tau_i) \le k'\tau'_i$. Since we want to apply Theorem 2.1, let $\lambda = s(\varepsilon, E[X']) = s(\varepsilon, k'\tau'_i)$ and $\sigma^2 = \mathrm{Var}[X']$.

First, consider the case that $\lambda = s(\varepsilon, k'\tau'_i) = \varepsilon \ln n$, i.e., that $(k'\tau'_i \ln n)^{1/2} \le \ln n$, which is equivalent to $k' \le (1/\tau'_i) \ln n$. Note that $\lambda^2/\sigma^2 \ge \varepsilon^2 \ln n \ge \ln n$, as $\varepsilon \ge 1$. Thus, $\min\{\lambda^2/\sigma^2, \lambda\} \ge \varepsilon \ln n$.

Now consider the case $\lambda = s(\varepsilon, k'\tau'_i) = \varepsilon(k'\tau'_i \ln n)^{1/2}$, i.e., that $(k'\tau'_i \ln n)^{1/2} \ge \ln n$, which is equivalent to $k' \ge (1/\tau'_i) \ln n$. We see that $\lambda \ge \varepsilon \ln n$ and $\lambda^2/\sigma^2 \ge \varepsilon^2 \ln n$. Hence, as before, we get $\min\{\lambda^2/\sigma^2, \lambda\} \ge \varepsilon \ln n$.

Combining both cases and applying Theorem 2.1, we get

$$\Pr[X' \ge k'\tau'_i + s(\varepsilon, k'\tau'_i)] = \Pr[X' \ge E[X'] + \lambda] \le e^{-\frac{1}{3}\min\left\{\frac{\lambda^2}{\sigma^2}, \lambda\right\}} \le e^{-\frac{\varepsilon}{3}\ln n} = n^{-\frac{\varepsilon}{3}}.$$

That is, the probability of detecting a (false) significance during k' trials is at most $n^{-\varepsilon/3}$. Since we look for a significance a total of at most $\log_2 k$ times during an iteration, we get by a union bound that the probability of detecting a significance within a history of length k is at most $n^{-\varepsilon/3} \log_2 k$. □

Lemma 3.1 bounds the probability of detecting a false significance within a single iteration if there is no bias in selection. The following corollary trivially bounds the probability of detecting a false significance within any number of iterations.

Corollary 3.2. *Consider the sig-cGA (Alg. 1) with $\varepsilon \ge 1$ running for k iterations such that, during each iteration, for each $i \in [n]$, a 1 is added to H_i with probability τ_i. Then the probability that at least one frequency will change during that time is at most $kn^{1-\varepsilon/3} \log_2 k$.*

Proof. For any $i \in [n]$ during any of the k iterations, by Lemma 3.1, the probability that τ_i changes is at most $n^{-\varepsilon/3} \log_2 k$. Via a union bound over all k iterations and all n frequencies, the statement follows. □

3.2 Efficient Implementation of the sig-cGA

In order to reduce the number of operations performed (computational cost) of the sig-cGA, we only check significance in historic data of lengths that are a power of 2. By saving the whole history but precomputing the number of 1s in the power-of-two intervals, a significance check can be done in time logarithmic in the history length; the necessary updates of this data structure can be done in logarithmic time (per bit-position) as well. With this implementation, the main loop of the sig-cGA has a computational cost of $O(\sum_{i=1}^{n} |H_i|)$. Since the histories are never longer than the run time (number of fitness evaluations; twice the number of iterations), we see that the computational cost is at most $O(nT \log T)$, when the run time is T. Since for most EAs working on bit string representations of length n the computational cost is larger than the run time by at least a factor of n, we see that our significance approach is not overly costly in terms of computational cost.

What appears unfavorable, though, is the memory usage caused by storing the full history. For this reason, we now sketch a way to condense the history so that it only uses space logarithmic in the length of the full history. This approach will not allow to access exactly the number of 1s (or 0s) in all power-of-two length histories. It will allow, however, for each $\ell \in [|H_i|]$, to access the number of 1s in some interval of length ℓ' with $\ell \le \ell' < 2\ell$. For reasons of readability, we shall in the subsequent analyses nevertheless regard the original sig-cGA, but it is quite clear that the mildly different accessibility of the history in the now-proposed condensed implementation will not change the asymptotic run times shown in this work.

For our condensed storage of the history, we have a list of blocks, each storing the number of 1s in some discrete interval $[t_1..t_2]$ of length equal to a power of two (including 1). When a new item has to be stored, we append a block of size 1 to the list. Then, traversing the list in backward direction, we check if there are three consecutive blocks of the same size, and if so, we merge the two earliest ones into a new block of twice the size. By this, we always maintain a list of blocks such that, for a certain power 2^k, there are between one and two blocks of length 2^j for all $j \in [0..k]$. This structural property implies both that we only have a logarithmic number of blocks (as we have $k = O(\log |H_i|)$) and that we can (in amortized constant time) access all historic intervals consisting of full blocks, which in particular implies that we can access an interval with length in $[2^j, 2^{j+1} - 1]$ for all $j \in [0..k]$.

3.3 Run Time Results for LeadingOnes and OneMax

We now prove our main results, that is, upper bounds of $O(n \log n)$ for the expected run time of the sig-cGA on LeadingOnes and OneMax. Note that the sig-cGA samples two offspring each iteration. Thus, up to a constant factor of 2, the expected run time is equal to the expected number of iterations until an optimum is sampled. In our proofs, we only consider the number of iterations.

We mention briefly that the sig-cGA is unbiased in the sense of Lehre and Witt [17], that is, it treats bit values and bit positions in a symmetric fashion. Consequently, all of our results hold not only for OneMax and LeadingOnes as defined in eq. (1) but also any similar function where an x_i may be changed to a $1 - x_i$ or swapped with an x_j (with $j \neq i$), as the sig-cGA has no bias for 1s or 0s, nor does it prefer certain positions over other positions. (In fact, it treats all positions exactly the same.)

In our proofs, we use the following lemma to bound probabilities split up by the law of total probability.

Lemma 3.3. *Let $\alpha, \beta, x, y \in \mathbb{R}$ such that $x \leq y$ and $\alpha \leq \beta$. Then $\alpha x + (1 - \alpha)y \geq \beta x + (1 - \beta)y$.*

We start with the expected run time on LeadingOnes.

Theorem 3.4. *Consider the sig-cGA (Alg. 1) with $\varepsilon > 12$ being a constant. Its run time on LeadingOnes is $O(n \log n)$ with high probability and in expectation.*

Proof. We start by showing that the run time is $O(n \log n)$ with high probability. Then we give a proof sketch of the expected run time. During this proof, we condition on the event that no frequency decreases during $O(n \log n)$ iterations, i.e., no (false) significance of 0s is detected. Note that, for any position $i \in [n]$, the probability of saving a 1 in H_i is at least τ_i, as the selection with respect to LeadingOnes has a bias for 1s. Thus, by Corollary 3.2, the probability that at least one frequency decreases during $O(n \log n)$ iterations is at most $O(n^{2-\varepsilon/3} \log^2 n)$, which is, as $\varepsilon > 12$, in $O(n^{-\varepsilon'})$, for an $\varepsilon' > 2$. Thus, with high probability, no frequency decreases during $O(n \log n)$ iterations.

The main idea of this proof is to show that the left-most frequency that is different from $1 - 1/n$ has a significant surplus of 1s in its history strong enough so that, after a logarithmic number of iterations, we change such a frequency from its initial value of 1/2 to $1 - 1/n$.

In order to make this idea precise, we now consider an iteration such that there is a frequency $\tau_i = 1/2$ such that, for all $j < i$, $\tau_j = 1 - 1/n$. We lower-bound the probability of saving a 1 in H_i in order to get an upper bound on the expected time until we detect the significance necessary to update τ_i to $1 - 1/n$. When considering position i, we assume an empty history although it is most likely not. We can do so, since the sig-cGA checks for a significance in different sub-histories of H_i (starting from the latest entry). Thus, we only consider sub-histories that go as far as the point in time when all indices less than i were at $1 - 1/n$.

Let O denote the event that we save a 1 this iteration, and let A denote the event that at least one of the two offspring during this iteration has a 0 at a position in $[i - 1]$. Note that event A means that the bit at position i of the winning individual is not relevant for selection. Hence, if A occurs, we save a 1 with probability $\tau_i = 1/2$. Otherwise, that is, the bit at position i is relevant for selection, we save a 1 with probability $1 - (1/2)^2 = 3/4$ (i.e., if we do not sample two 0s). Formally, $\Pr[O] = \Pr[A] \cdot \frac{1}{2} + \Pr[\overline{A}] \cdot \frac{3}{4}$, which is a convex combination of 1/2 and 3/4. Thus, according to Lemma 3.3, we get a lower bound if we decrease the factor of the larger term, namely, $\Pr[\overline{A}]$. The event $\overline{A}$ occurs if and only if both offspring have only 1s at the positions 1 through $i - 1$: $\Pr[\overline{A}] = (1 - 1/n)^{2(i-1)}$, as we assumed that all frequencies at indices less than i are already at $1 - 1/n$. Note that this term is minimal for $i = n$. Thus, we get $\Pr[\overline{A}] \geq e^{-2}$ by using the well-known inequality $(1 - 1/n)^{n-1} \geq e^{-1}$. Overall, we get $\Pr[O] \geq (1 - e^{-2}) \cdot \frac{1}{2} + e^{-2} \cdot \frac{3}{4} = \frac{1}{2}\left(1 + \frac{1}{2}e^{-2}\right)$.

Let $X \sim \mathrm{Bin}\big(k, (1/2)(1 + e^{-2}/2)\big)$ denote a random variable that is stochastically dominated by the real process of saving 1s at position i. In order to get a bound on the number of iterations k that we need for detecting a significance of 1s, we bound the probability of a significance not occurring in a history of length k, i.e., we save less than $k/2 + s(3\varepsilon, k/2)$ 1s:

$$\Pr\left[X < \frac{k}{2} + s\left(\varepsilon, \frac{k}{2}\right)\right] \leq \Pr\left[X \leq E[X] - \left(\frac{k}{4}e^{-2} - s\left(\varepsilon, \frac{k}{2}\right)\right)\right],$$

where the minuend is positive if $ke^{-2}/4 > s(\varepsilon, k/2)$, which is the case for $k > 8e^4\varepsilon^2 \ln n > \ln n$, since we assume that $\varepsilon > 12$. Let $c = 8e^4\varepsilon^2$. For $k \geq 4c \ln n$, we get that $ke^{-2}/4 - s(\varepsilon, k/2) \geq ke^{-2}/8 =: \lambda$. By applying Theorem 2.1 for any $k \geq 4c \ln n$ and noting that $\mathrm{Var}[X] \leq k/4$ and, thus, $\lambda^2/\mathrm{Var}[X] \leq \lambda$, we get

$$\Pr\left[X < \frac{k}{2} + s\left(\varepsilon, \frac{k}{2}\right)\right] \leq \Pr\left[X \leq E[X] - \frac{k}{8}e^{-2}\right]$$
$$\leq e^{-\frac{1}{3} \cdot \frac{4k^2}{64k} e^{-4}} = e^{-\frac{k}{48} e^{-4}} \leq n^{-\frac{c}{12} e^{-4}} = n^{-\frac{\varepsilon^2}{3}}.$$

Thus, with probability at least $1 - n^{-\varepsilon^2/3}$, the τ_i will be set to $1 - 1/n$ after $8e^4\varepsilon^2 \ln n = O(\log n)$ iterations. Further, via a union bound over all n frequencies, the probability of any such frequency not being updated to $1 - 1/n$ after $O(\log n)$ iterations is at most $n^{1-\varepsilon^2/3} \leq n^{-47}$, as $\varepsilon > 12$. Hence, with high probability, all frequencies will be set to $1 - 1/n$.

Taking together the results of all frequencies being updated to $1 - 1/n$, each in time $O(\log n)$, and no frequency at 1/2 or $1 - 1/n$ decreasing, all with high probability, yields that all frequencies are at $1 - 1/n$ within $O(n \log n)$ iterations. Then the optimum is sampled with probability $(1 - 1/n)^n \geq 1/(2e) = \Omega(1)$, i.e., with constant probability. Hence, we have to wait $O(\log n)$ additional iterations in order to sample the optimum with high probability.

As for the expected run time, we are left to bound the expected time if a frequency decreases in the initial $O(n \log n)$ iterations, which only happens with a probability of $O(n^{-\varepsilon'})$. Due to Lemma 3.1 and similar to Corollary 3.2, during t iterations and considering an interval of length t', no frequency decreases with a probability of at least $1 - t' n^{1-\varepsilon/3} \log_2 t$. By assuming $t \leq n^{2n}$ and $t' = \Theta(n^2 \log n)$, with high probability, no frequency decreases during such an interval, as $\varepsilon > 12$. By analogous calculations as done for a leftmost frequency at $1/2$, it can be shown that a leftmost frequency at $1/n$ is increased to during $\Theta(n \log n)$ iterations with high probability. Thus, with high probability, the sig-cGA finds the optimum during an interval of length t'. If not, we repeat this argument until $t > n^{2n}$. Then, the algorithm is expected to have found the optimum by pessimistically assuming that all frequencies are at $1/n$. □

For our next result, we make use of the following lemma based on a well-known estimate of binomial coefficients close to the center. A proof was given by, e.g., Doerr and Winzen [8]. We use it to show how likely it is that two individuals sampled from the sig-cGA have the same OneMax value.

Lemma 3.5. *For $c \in \Theta(1)$, $\ell \in \mathbb{N}^+$, let $k \in [\ell/2 \pm c\sqrt{\ell}]$ and let $X \sim \mathrm{Bin}(1/2, \ell)$. Then $\Pr[X = k] = \Omega(1/\sqrt{\ell})$.*

The next theorem shows that the sig-cGA is also able to optimize OneMax within the same asymptotic time like many other EAs.

Theorem 3.6. *Consider the sig-cGA (Alg. 1) with $\varepsilon > 12$ being a constant. Its run time on OneMax is $O(n \log n)$ with high probability and in expectation.*

Proof. As in the proof of Theorem 3.4, we start by showing that the run time holds with high probability. For this, we condition on the event that no frequency decreases during $O(n \log n)$ iterations. This can be argued in the same way as in the aforementioned proof.

The main idea of this proof is to show that, for any frequency at $1/2$, $O(n \log n)$ iterations are enough in order to detect a significance in 1s. This happens in parallel for all frequencies. For our argument to hold, it is only important that all the other frequencies are at $1/2$ or $1 - 1/n$, which we condition on.

Formally, during any of the $O(n \log n)$ iterations, let $\ell \in [n]$ denote the number of frequencies at $1/2$. Then $n - \ell$ frequencies are at $1 - 1/n$. Further, consider a position $i \in [n]$ with $\tau_i = 1/2$. We show that such a position will sample 1s significantly more often than the hypothesis by a factor of $\Theta(1/\sqrt{\ell})$. Then τ_i will be updated to $1 - 1/n$ within $O(\ell \log n)$ iterations.

In order to show that 1s are significantly more often saved than assumed, we proceed as follows: we consider that all bits but bit i of both offspring during any iteration have been sampled. If the number of 1s of both offspring differs by more than one, bit i cannot change the outcome of the selection process – bit i will be 1 with probability $1/2$. However, if the number of 1s differs by at most one, then the outcome of bit i in both offspring has an influence on whether a 1 is saved or not, i.e., this introduces a bias toward saving a significant amount of 1s.

Let O denote the event to save a 1 at position i this iteration, and let A denote the event that the numbers of 1s (excluding position i) of both offspring differ by at least two during that iteration. Then the probability to save a 1, conditioned on A, is $1/2$.

In the case of $\overline{A}$, we make a case distinction with respect to the absolute difference of the number of 1s of both offspring, excluding position i. If the difference is zero, then a 1 will be saved if not both offspring sample a 0, which happens with probability $1 - (1/2)^2 = 3/4$. If the absolute difference is one, then a 1 will be saved if the winner (with respect to all bits but bit i) samples a 1 (with probability $1/2$) or if it samples a 0, the loser samples a 1, and the loser is chosen during selection, which happens with probability $(1/2)^3 = 1/8$. Overall, the probability that a 1 is saved is at least $1/2 + 1/8 = 5/8$ in the case of $\overline{A}$.

Combining both cases, we see that $\Pr[O] \geq \Pr[A] \cdot \frac{1}{2} + \Pr\left[\overline{A}\right] \cdot \frac{5}{8}$, which can be lower-bounded by determining a lower bound for $\Pr\left[\overline{A}\right]$, according to Lemma 3.3.

With respect to $\Pr\left[\overline{A}\right]$, we first note that the probability that the $n - \ell$ frequencies at $1 - 1/n$ will all sample a 1 for both offspring is $(1 - 1/n)^{2(n-\ell)} \geq e^{-2}$, as $n - \ell \leq n - 1$. Now we only consider the difference of 1s sampled with respect to $\ell' := \ell - 1$, for $\ell \geq 2$, positions with frequencies at $1/2$, i.e., all remaining positions but i. Since all of the respective frequencies are at $1/2$, the expected number of 1s is $\ell'/2$. Due to Theorem 2.1 (or, alternatively, Chebyshev's inequality), the probability of deviating from this value by more than $\sqrt{\ell'/2}$ is at most a constant $c < 1$. Conditional on sampling a number of 1s in the range of $\ell'/2 \pm \sqrt{\ell'/2}$, the probability to sample $k \in [\ell'/2 \pm \sqrt{\ell'/2}]$ 1s is, due to Lemma 3.5, $\Omega(1/\sqrt{\ell'})$, since all ℓ' frequencies are at $1/2$. Thus, by law of total probability, the probability that both offspring have the same number of 1s or differ only by one, i.e., $\Pr\left[\overline{A}\right]$, is, for a constant $d > 0$, at least $d/\sqrt{\ell'}$. Hence, we get, for a sufficiently small constant $d' > 0$, factoring in the probability of $1 - c$ of the number of 1s being concentrated around $\ell'/2$ and the remaining $n - \ell$ positions only sampling 1s,

$$\Pr[O] \geq \left(1 - e^{-2}(1-c)\frac{d}{\sqrt{\ell'}}\right) \cdot \frac{1}{2} + e^{-2}(1-c)\frac{d}{\sqrt{\ell'}} \cdot \frac{5}{8} \geq \frac{1}{2} + \frac{d'}{\sqrt{\ell}} .$$

This means that the sig-cGA expects 1s to occur with probability $1/2$, but they occur with a probability of at least $1/2 + d'/\sqrt{\ell}$. Note that for the case $\ell = 1$, i.e., $\ell' = 0$, conditional on the remaining $n - \ell$ positions only sampling 1s, $\Pr\left[\overline{A}\right] = 1$ and hence $\Pr[O] \geq (1 - e^{-2}) \cdot 1/2 + e^{-2} \cdot 5/8$. Thus, we use $1/2 + d'/\sqrt{\ell}$ as a lower bound for $\Pr[O]$ in all cases for ℓ, for an appropriately chosen d'.

Analogous to the proof of Theorem 3.4, let $X \sim \mathrm{Bin}\left(k, 1/2 + d'/\sqrt{\ell}\right)$ denote a random variable that is stochastically dominated by the real process of saving 1s at position i. We bound the probability of not detecting a significance of 1s after k iterations, i.e.,

$$\Pr\left[X < \frac{k}{2} + s\left(\varepsilon, \frac{k}{2}\right)\right] \leq \Pr\left[X \leq E[X] - \left(\frac{kd'}{\sqrt{\ell}} - s\left(\varepsilon, \frac{k}{2}\right)\right)\right] .$$

Let $k \geq 2(\varepsilon^2/d'^2)\ell \ln n$. Then $kd'/\sqrt{\ell} - s(\varepsilon, k/2) \geq kd'/(2\sqrt{\ell}) =: \lambda$. By noting that $\mathrm{Var}[X] \leq k/4$ and $\lambda^2/\mathrm{Var}[X] \leq \lambda$ for d' sufficiently small, we get by applying Theorem 2.1,

$$\Pr\left[X < \frac{k}{2} + s\left(\varepsilon, \frac{k}{2}\right)\right] \leq \Pr\left[X \leq E[X] - \frac{kd'}{2\sqrt{\ell}}\right]$$
$$\leq e^{-\frac{1}{3} \cdot \frac{4k^2 d'^2}{4\ell k}} = e^{-\frac{kd'^2}{3\ell}} \leq e^{-\frac{2}{3}\varepsilon^2 \ln n} = n^{-\frac{2}{3}\varepsilon^2} .$$

Thus, with probability at least $1 - n^{-2\varepsilon^2/3}$, τ_i will be set to $1 - 1/n$ after $2(\varepsilon^2/d'^2)\ell \ln n = O(\ell \log n)$ iterations. Further, via a union bound over all n frequencies, the probability of any such frequency not being updated to $1 - 1/n$ after $O(\ell \log n)$ iterations is at most $n^{1-2\varepsilon^2/3} \leq n^{-95}$, as $\varepsilon > 12$. Hence, with high probability, all frequencies will be set to $1 - 1/n$.

Since our argument for position i was made for an arbitrary i and independent of the other positions (except that their frequencies are either $1/2$ or $1 - 1/n$), and since all n frequencies start at $1/2$ (i.e., $\ell = n$), we have to wait at most $O(n \log n)$ iterations until all frequencies are set to $1 - 1/n$ with high probability. Then, with a probability of at least $(1 - 1/n)^n \geq 1/(2e) = \Omega(1)$, the optimum will be sampled. Hence, after $O(\log n)$ additional iterations, the optimum will be sampled with high probability.

The expected run time can be proven similarly as argued in the proof of Theorem 3.4. The main difference here is that, assuming all frequencies are at $1/n$, with high probability, all frequencies will increase during $O(n^2 \log n)$ iterations (in parallel, not sequentially). Further, since $\varepsilon > 12$, no frequency will decrease during an interval of such length with high probability. □

Note that although the expected run time of the sig-cGA is asymptotically the same on LeadingOnes and OneMax, the reason is quite different: for LeadingOnes, the sig-cGA sets its frequencies quickly consecutively to $1-1/n$, as it only needs $O(\log n)$ iterations per frequency in expectation. This is due to the bias for saving 1s being very large (constant, in fact) when all frequencies to the left are at $1 - 1/n$, i.e., when it is very likely that bit i is relevant for selection. Friedrich et al. [11] exploit this fact in the analysis (and design) of the scGA heavily, which is why it, too, has an expected run time of $O(n \log n)$ on LeadingOnes. However, when not all frequencies to the left of a position are at $1 - 1/n$, the bias is almost negligible, as it is necessary that bits sampled with frequencies of at most $1/2$ have to sample the same value. Thus, in this case, the probability of this happening declines exponentially in the number of frequencies to the left not being at $1 - 1/n$.

For OneMax, the situation is different. The bias in selection only gets strong (i.e., increases by a constant additive term) when a constant number of frequencies is left at $1/2$ and has not reached $1-1/n$. More general, when ℓ frequencies are still at $1/2$, the bias only adds a term of roughly $1/\sqrt{\ell}$. Thus, it takes longer in expectation in order to detect a significance for a position. However, the bias is constantly there and, even for $\ell = n$, very large when compared to the bias for LeadingOnes for a position whose frequencies to the left are not all at $1-1/n$. Hence, for OneMax, the frequencies can be increased in parallel. This is the major difference to LeadingOnes, where the frequencies are increased sequentially.

4 RUN TIME ANALYSIS FOR THE SCGA

Being the closest competitor to the sig-cGA in that it also optimizes LeadingOnes in $O(n \log n)$ in expectation is the *stable compact genetic algorithm* (scGA; Alg. 2), which is a variant of the cGA [13] and was introduced by Friedrich et al. [11] in order to present an EDA that optimizes LeadingOnes in time $O(n \log n)$. It works very similar to the cGA, however, it introduces a bias toward the update that favors frequencies moving to $1/2$. For this purpose, the scGA has, next to the parameter ρ of the cGA, another parameter $a \in O(\rho)$, which works in the following way: when a frequency above $1/2$ is decreased, it decreases by $\rho + a$, not only by ρ as in the case of the cGA. However, a frequency above $1/2$ is still only increased by ρ. For a frequency below $1/2$, this is done analogously.

Algorithm 2: The scGA [11] with parameters ρ, a, and d optimizing f

```
t ← 0;
for $i \in [n]$ do $\tau_i^{(t)} \leftarrow \frac{1}{2}$;
repeat
    $x, y \leftarrow$ offspring sampled with respect to $\tau^{(t)}$;
    $(x, y) \leftarrow$ winner/loser of $x$ and $y$ with respect to $f$;
    for $i \in [n]$ do
        if $x_i > y_i$ then
            if $\tau_i^{(t)} \leq \frac{1}{2}$ then $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)} + \rho + a$;
            else if $\frac{1}{2} < \tau_i^{(t)} < d$ then $\tau^{(t+1)} \leftarrow \tau_i^{(t)} + \rho$;
            else $\tau_i^{(t+1)} \leftarrow 1$;
        else if $x_i < y_i$ then
            if $\tau_i^{(t)} \geq \frac{1}{2}$ then $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)} - \rho - a$;
            else if $1 - d < \tau_i^{(t)} < \frac{1}{2}$ then $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)} - \rho$;
            else $\tau_i^{(t+1)} \leftarrow 0$;
        else $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)}$;
    $t \leftarrow t + 1$;
until termination criterion met;
```

Further, the scGA has a third parameter $d \in (1/2, 1)$, which marks the borders for a frequency that are sufficient in order to set it to one of its extreme values, i.e., 0 or 1. If a frequency τ_i is greater or equal to d, it is updated to 1 and can then never be changed again, as all bits at position i will be 1s. Symmetrically, if a frequency τ_i is less or equal to $1 - d$, it is updated to 0. Intuitively, the parameter d describes a significance value that is sufficient for the algorithm to fully commit for a bit value.

The intention of the scGA is that each frequency stays around $1/2$ as long as there is no strong bias toward either bit value for its respective position. Once the bias is strong enough, the algorithm is willing to fix the bits for that position. While this approach works well when there is a strong bias in a position (i.e., as in LeadingOnes [11]), it fails when the bias is only weak (i.e., as in OneMax; Thm. 4.1).

We prove that the scGA is not able to optimize OneMax as fast as the sig-cGA, as it is not able to detect the comparably small bias of $1/\sqrt{n}$ for OneMax when compared to the strong bias of $\Theta(1)$ for LeadingOnes for a frequency whose frequencies to the left are at $1 - 1/n$. Note that the assumptions in Theorem 4.1 for ρ and d are similar to the ones used by Friedrich et al. [11] in order to prove the expected run time of $O(n \log n)$ of the scGA on LeadingOnes. Our assumption for a is more restrictive, as we require $a = \Theta(\rho)$, whereas Friedrich et al. [11] only require $a = O(\rho)$. However, we allow $\rho = O(1/\log n)$, whereas Friedrich et al. [11] require $\rho = \Theta(1/\log n)$.

Theorem 4.1. *Let $\alpha \in (0, 1]$ be a constant. Consider the scGA (Alg. 2) with $\rho = O(1/\log n)$, $a = \alpha\rho$, and $1/2 < d \leq 5/6$ with $d =$*

$\Theta(1)$. *Its run time on* OneMax *is* $\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ *in expectation and with high probability for a constant* $c > 0$.

Proof sketch of Thm. 4.1. We show that within $O\big(2^{c/\rho}\big)$ iterations, all frequencies of the scGA will still stay in the interval $(1 - d, d)$. That is, each frequency is still a constant away from either 1 or 0. This means it is exponentially unlikely to sample the optimum until then. Thus, the expected run time is at least $\Omega\big(2^{c/\rho}\big)$.

Our proof is mostly an application of the negative drift theorem by Oliveto and Witt [20, 21]. In order to calculate the drift, i.e., the expected change of a frequency in a single iteration, it is necessary that all frequencies are still in the interval $(1 - d, d)$. Following a result by Sudholt and Witt [23], the bias in selection is then only in the order of $O(\rho/\sqrt{n})$, which is too little compared to the artificial bias in the update, which is in the order of $a = \Theta(\rho)$. Thus, we get a first-hitting time of a frequency reaching at least d that is exponential in $1/\rho$.

The second bound of $2^{\Theta(n)}$ follows by the constant chance of sampling the optimum with a probability of at most $(5/6)^n$. □

5 CONCLUSIONS

We introduced a new algorithm (sig-cGA) that is able to optimize both OneMax and LeadingOnes in time $O(n \log n)$ with high probability and in expectation, which is the first result of this kind for an EDA or even an EA. The sig-cGA achieves these run times by only performing an update to its stochastic model once it notices a significance in its history of samples. In contrast to that, typical theoretically investigated EDAs or EAs do not save the entire history of samples but only a small part thereof: EAs save some samples in their population whereas EDAs store information about the history implicitly in their frequency vector.

Since it is quite memory-consuming to store all samples seen so far the longer the sig-cGA runs, we proposed a way of efficiently saving all of the necessary information for the algorithm, which is the number of 1s or 0s seen so far. Currently, the sig-cGA saves new information every iteration. However, whenever both offspring sample the same value, the algorithm does not learn anything. Thus, an even more memory-efficient approach would be to only save a bit value if the one of the winning offspring differs from the respective bit value of the loser. This is how the cGA actually performs an update. However, since the intention of the sig-cGA is to keep its frequencies as long as possible at 1/2 until it detects a (hopefully correct) significance, this approach reduces the memory necessary only by a constant factor of 2, due to classical Chernoff bounds.

Overall, the sig-cGA trades slightly increased memory (due to its history) for reduced run times, which is a very good payoff.

ACKNOWLEDGMENTS

This work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, LabEx LMH, in a joint call with Gaspard Monge Program for optimization, operations research and their interactions with data sciences. This publication is based upon work from COST Action CA15140, supported by COST.

REFERENCES

[1] Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, and Kurt Mehlhorn. 2013. The query complexity of finding a hidden permutation. In *Space-Efficient Data Structures, Streams, and Algorithms (Lecture Notes in Computer Science)*, Vol. 8066. Springer, 1–11. Full version available online at http://eccc.hpi-web.de/report/2012/087/.
[2] Gautham Anil and R. Paul Wiegand. 2009. Black-box search by elimination of fitness functions. In *Proc. of FOGA'09*. ACM, 67–78. DOI: http://dx.doi.org/10.1145/1527125.1527135
[3] Duc-Cuong Dang and Per Kristian Lehre. 2015. Simplified runtime analysis of estimation of distribution algorithms. In *Proc. of GECCO'15*. ACM, 513–518. DOI: http://dx.doi.org/10.1145/2739480.2754814
[4] Benjamin Doerr. 2018. Probabilistic Tools for the Analysis of Randomized Optimization Heuristics. *ArXiv e-prints* (2018). arXiv:1801.06733
[5] Benjamin Doerr and Carola Doerr. 2015. A tight runtime analysis of the (1+(λ, λ)) genetic algorithm on OneMax. In *Proc. of GECCO'15*. ACM, 1423–1430. DOI: http://dx.doi.org/10.1145/2739480.2754683
[6] Benjamin Doerr and Marvin Künnemann. 2015. Optimizing linear functions with the (1+λ) evolutionary algorithm – different asymptotic runtimes for different instances. *Theoretical Computer Science* 561 (2015), 3–23. DOI: http://dx.doi.org/10.1016/j.tcs.2014.03.015
[7] Benjamin Doerr, Frank Neumann, Dirk Sudholt, and Carsten Witt. 2007. On the runtime analysis of the 1-ANT ACO algorithm. In *Proc. of GECCO'07*. ACM, 33–40. DOI: http://dx.doi.org/10.1145/1276958.1276964
[8] Benjamin Doerr and Carola Winzen. 2014. Ranking-based black-box complexity. *Algorithmica* 68 (2014), 571–609. DOI: http://dx.doi.org/10.1007/s00453-012-9684-9
[9] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the analysis of the (1+1) evolutionary algorithm. *Theoretical Computer Science* 276, 1-2 (2002), 51–81. DOI: http://dx.doi.org/10.1016/S0304-3975(01)00182-7
[10] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2006. Upper and lower bounds for randomized search heuristics in black-box optimization. *Theory of Computing Systems* 39 (2006), 525–544.
[11] Tobias Friedrich, Timo Kötzing, and Martin S. Krejca. 2016. EDAs cannot be balanced and stable. In *Proc. of GECCO'16*. 1139–1146. DOI: http://dx.doi.org/10.1145/2908812.2908895
[12] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, and Andrew M. Sutton. 2017. The compact genetic algorithm is efficient under extreme Gaussian noise. *IEEE Transactions on Evolutionary Computation* 21, 3 (2017), 477–490.
[13] Georges R. Harik, Fernando G. Lobo, and David E. Goldberg. 1999. The compact genetic algorithm. *IEEE Transactions on Evolutionary Computation* 3, 4 (1999), 287–297.
[14] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. 2005. On the choice of the offspring population size in evolutionary algorithms. *Evolutionary Computation* 13 (2005), 413–440. DOI: http://dx.doi.org/10.1162/106365605774666921
[15] Martin S. Krejca and Carsten Witt. 2017. Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax. In *Proc. of FOGA'17*. ACM, 65–79. DOI: http://dx.doi.org/10.1145/3040718.3040724
[16] Per Kristian Lehre and Phan Trung Hai Nguyen. 2017. Improved runtime bounds for the univariate marginal distribution algorithm via anti-concentration. In *Proc. of GECCO'17*. ACM, 1383–1390. DOI: http://dx.doi.org/10.1145/3071178.3071317
[17] Per Kristian Lehre and Carsten Witt. 2012. Black-box search by unbiased variation. *Algorithmica* 64, 4 (2012), 623–642. DOI: http://dx.doi.org/10.1007/s00453-012-9616-8
[18] Alberto Moraglio and Dirk Sudholt. 2017. Principled design and runtime analysis of abstract convex evolutionary search. *Evolutionary Computation* 25, 2 (2017), 205–236. DOI: http://dx.doi.org/10.1162/EVCO_a_00169
[19] Frank Neumann and Carsten Witt. 2009. Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica* 54, 2 (2009), 243–255. DOI: http://dx.doi.org/10.1007/s00453-007-9134-2
[20] Pietro S. Oliveto and Carsten Witt. 2011. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica* 59, 3 (2011), 369–386. DOI: http://dx.doi.org/10.1007/s00453-010-9387-z
[21] Pietro S. Oliveto and Carsten Witt. 2012. Erratum: simplified drift analysis for proving lower bounds in evolutionary computation. *CoRR* abs/1211.7184 (2012). http://arxiv.org/abs/1211.7184
[22] Martin Pelikan, Mark Hauschild, and Fernando G. Lobo. 2015. Estimation of distribution algorithms. In *Springer Handbook of Computational Intelligence*. 899–928. DOI: http://dx.doi.org/10.1007/978-3-662-43505-2_45
[23] Dirk Sudholt and Carsten Witt. 2016. Update strength in EDAs and ACO: how to avoid genetic drift. In *Proc. of GECCO'16*. 61–68. DOI: http://dx.doi.org/10.1145/2908812.2908867
[24] Carsten Witt. 2006. Runtime analysis of the (μ + 1) EA on simple pseudo-Boolean functions. *Evolutionary Computation* 14 (2006), 65–86. DOI: http://dx.doi.org/10.1162/evco.2006.14.1.65
[25] Carsten Witt. 2017. Upper bounds on the runtime of the univariate marginal distribution algorithm on OneMax. In *Proc. of GECCO'17*. ACM, 1415–1422. DOI: http://dx.doi.org/10.1145/3071178.3071216
[26] Zijun Wu, Michael Kolonko, and Rolf H. Möhring. 2017. Stochastic runtime analysis of the cross-entropy algorithm. *IEEE Transactions on Evolutionary Computation* 21, 4 (2017), 616–628. DOI: http://dx.doi.org/10.1109/TEVC.2017.2667713