Significance-based estimation-of-distribution algorithms

Benjamin Doerr, Martin Krejca

Significance-based Estimation-of-Distribution Algorithms

Benjamin Doerr* and Martin S. Krejca†

Abstract

Estimation-of-distribution algorithms (EDAs) are randomized search heuristics that create a probabilistic model of the solution space, which is updated iteratively, based on the quality of the solutions sampled according to the model. As previous works show, this iteration-based perspective can lead to erratic updates of the model, in particular, to bit-frequencies approaching a random boundary value. In order to overcome this problem, we propose a new EDA based on the classic compact genetic algorithm (cGA) 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 commonly regarded benchmark functions OneMax, LeadingOnes, and BinVal all in quasilinear 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 its hypothetical population size. Similarly, we show that the convex search algorithm cannot optimize OneMax in polynomial time.

1 Introduction

Estimation-of-distribution algorithms (EDAs; [40]) are nature-inspired heuristics, similar to *evolutionary algorithms* (EAs). In contrast to EAs, which maintain an explicit set of solutions, EDAs optimize a function by evolving a probabilistic model of the solution space. Iteratively, an EDA uses its probabilistic model in order to generate samples and make observations from them. It then updates its model based on these observations, where an algorithm-specific parameter determines how strong the changes to the model in each iteration are.

For an EDA to succeed in optimization, it is important that its model is changed over time in a way 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.

*Laboratoire d'Informatique (LIX), École Polytechnique, Palaiseau, France
e-mail: doerr@lix.polytechnique.fr
†Hasso Plattner Institute, University of Potsdam, Potsdam, Germany
e-mail: martin.krejca@hpi.de

The theoretical analysis of EDAs has recently gained momentum (see, e.g., the survey [31]) and has clearly shown that this trade-off between convergence speed and accumulation of erratic updates can be delicate and non-trivial to understand. Among the most relevant works, Sudholt and Witt [41] and Krejca and Witt [30] 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 update parameter for the model is too large, the model converges too quickly and very likely to a wrong model; consequently, it then takes a long time to find the optimum. On the other hand, if the parameter is too small, then the model converges to the correct model but does so slowly. More formally, Sudholt and Witt [41] prove a lower bound of $\Omega(K\sqrt{n} + n \log n)$ for the 2-MMAS$_{\text{ib}}$ and the cGA, where $1/K$ is the step size of the algorithm, and Krejca and Witt [30] prove a lower bound of $\Omega(\lambda\sqrt{n} + n \log n)$ for the UMDA, where λ is the population size of the algorithm. These results show that choosing the parameter with a value of $\omega(\sqrt{n} \log n)$ has no benefit. Further, it has been recently shown by Lengler et al. [34] that the run time of the cGA on OneMax is $\Omega(K^{1/3}n + n \log n)$ for $K = O(\sqrt{n}/(\log n \log\log n))$. Together with the results from Sudholt and Witt [41], this implies a bimodal behavior in the run time with respect to K if $K = \Omega(\log n) \cap O(\sqrt{n} \log n)$, showing that the run time is sensitive to the parameter choice.

Friedrich et al. [25] also discuss the problem of how to choose the update strength. They consider a class of EDAs optimizing functions over bit strings of length n that all current theoretical results fall into, named n-Bernoulli-λ-EDA. The model of such EDAs uses one variable per bit of a bit string, resulting in a probability vector 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.

Friedrich et al. [25] consider two different properties of such EDAs, namely *balanced* and *stable*. Intuitively, a *balanced* EDA does not change a frequency τ_i in expectation if the fitness function has no preference for 0s or 1s at position i. A *stable* EDA keeps a frequency, in such a scenario, close to 1/2. Friedrich et al. [25] then prove that an n-Bernoulli-λ-EDA cannot be both balanced and stable. They also prove that all commonly theoretically analyzed EDAs are balanced. This means that the frequencies will always move toward 0 or 1, even if there is no bias from the objective function.

Motivated by these results, Friedrich et al. [25] propose an EDA (called *scGA*) that is stable (but not balanced) by introducing an artificial bias into the update process that should counteract the bias of a balanced EDA. However, we prove that this approach fails badly on the standard benchmark function OneMax (Thm. 4). We note that a similar bias towards the middle frequency of 1/2 was proven for a binary differential evolution algorithm by Zheng et al. [44]. Similar to the situation of the scGA, their run time results (partially relying on mean-field assumptions) indicate that LeadingOnes is optimized in a number of generations that is linear in the problem size n. This gives an $O(n \log n)$ number of function evaluations when using a logarithmic population size (and smaller population sizes are provably not successful). For OneMax, the results are less conclusive, but they indicate a run time exponential in the population size can occur.

Table 1: Expected run times (number of fitness evaluations) of various algorithms until they first find an optimum for the two functions OM (eq. (1)) and LO (eq. (2)). For optimal parameter settings, many algorithms have a run time of $\Theta(n \log n)$ for OM and of $\Theta(n^2)$ for LO. We note that the $\big(1+(\lambda,\lambda)\big)$ GA has an $o(n \log n)$ run time on OM (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 LO.

Algorithm	OM	constraints	LO	constraints
$(1+1)$ EA	$\Theta(n \log n)$ [23]	none	$\Theta(n^2)$ [23]	none
$(\mu+1)$ EA	$\Theta(\mu n + n \log n)$ [42]	$\mu = O\big(\text{poly}(n)\big)$	$\Theta(\mu n \log n + n^2)$ [42]	$\mu = O\big(\text{poly}(n)\big)$
$(1+\lambda)$ EA	$\Theta\Big(n \log n + \frac{\lambda n \log\log\lambda}{\log\lambda}\Big)$ [14, 28][1]	$\lambda = O(n^{1-\varepsilon})$	$\Theta(n^2 + \lambda n)$ [28]	$\lambda = O\big(\text{poly}(n)\big)$
$(\mu+\lambda)$ EA	$\Theta\Big(\frac{n \log n}{\lambda} + \frac{n}{\lambda/\mu} + \frac{n \log^+\log^+\lambda/\mu}{\log^+\lambda/\mu}\Big)$ [4]	$\log^+ x := \max\{1, \log x\}$	$\Omega\Big(n^2 + \frac{\lambda n}{\log(\lambda/n)}\Big)$ [5]	–
$\big(1+(\lambda,\lambda)\big)$ GA	$\Theta\Big(\max\Big\{\frac{n \log n}{\lambda}, \frac{n\lambda \log\log\lambda}{\log\lambda}\Big\}\Big)$ [9]	$p = \frac{\lambda}{n}$, $c = \frac{1}{\lambda}$	unknown	–
CSA	$\Omega(n^c)$ [Thm. 6]	$c > 0$	$O(n \log n)$ [35]	$\mu \geq 8 \ln\big((4n+6)n\big)$, restarts
UMDA/PBIL[2]	$\Omega(\lambda\sqrt{n} + n \log n)$ [30]	$\mu = \Theta(\lambda)$	$O(n\lambda \log\lambda + n^2)$ [6, 32]	$\lambda = \Omega(\log n)$, $\mu = \Theta(\lambda)$
	$O(\lambda n)$ [6, 43]	$\mu = \Omega(\log n) \cap O(\sqrt{n})$, $\lambda = \Omega(\mu)$ or		
		$\mu = \Omega(\sqrt{n} \log n)$, $\mu = \Theta(\lambda)$ or		
		$\mu = \Omega(\log n) \cap o(n)$, $\mu = \Theta(\lambda)$		
cGA/2-MMAS$_{\text{ib}}$	$\Omega\Big(\frac{\sqrt{n}}{\rho} + n \log n\Big)$ [41]	$\frac{1}{\rho} = O\big(\text{poly}(n)\big)$	unknown	–
	$O\Big(\frac{\sqrt{n}}{\rho}\Big)$ [41]	$\frac{1}{\rho} = \Omega(\sqrt{n} \log n) \cap O\big(\text{poly}(n)\big)$		
1-ANT	$O(n^2)$ [37][3]	$\rho = \Omega(n^{-1+\varepsilon})$	$O(n^2 \cdot (6e)^{1/(n\rho)})$ [19]	none
			$2^{\Omega(\min\{n, 1/(n\rho)\})}$ [19]	none
MMAS*	$O\big(\frac{n \log n}{\rho}\big)$ [36]	$\rho = O(1)$	$O\big(n^2 + \frac{n \log n}{\rho}\big)$ [36]	$\rho = O(1)$
			$\Omega\big(n^2 + \frac{n}{-\rho \log(2\rho)}\big)$ [36]	$\rho = 1/\text{poly}(n)$
scGA	$\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ [Thm. 4]	$1/\rho = \Omega(\log n)$, $a = \Theta(\rho)$, $d = \Theta(1)$, $c > 0$	$O(n \log n)$ [25]	$1/\rho = \Theta(\log n)$, $a = O(\rho)$, $d = \Theta(1)$
sig-cGA (Alg. 1)	$O(n \log n)$ [Thm. 3]	$\varepsilon > 12$	$O(n \log n)$ [Thm. 2]	$\varepsilon > 12$

[1] Better run time bounds for the $(1+\lambda)$ EA are known if the mutation rate is (i) fitness-dependent [5], (ii) self-adjusting [11], or (iii) self-adaptive [21].
[2] The results shown for PBIL are the results of UMDA if not mentioned otherwise, since the latter is a special case of the former.
[3] For $\rho \geq (n-2)/(3n-2)$, the algorithm simulates the $(1+1)$ EA and has a run time of $\Theta(n \log n)$.

The results of Friedrich et al. [25], Sudholt and Witt [41], Krejca and Witt [30], and Lengler et al. [34] 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 probabilistic 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 balance-property may not be overcome, resulting in the probabilistic model converging to an incorrect one, as the algorithms are not stable. Hence, the parameter has to be chosen sufficiently small in order to guarantee convergence to the correct model, resulting in a slower optimization time.

As we shall argue in this work, the reason for this dilemma is that EDAs only use information from a single iteration when performing an update. Thus, the decision of whether and how a frequency should be changed has to be made on the spot, which may result in harmful decisions.

To overcome these difficulties, we propose a conceptually new EDA that has some access to the search history and updates the model only if there is sufficient reason. The *significance-based compact genetic algorithm* (sig-cGA) stores for each position the history of bits of good solutions 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 LeadingOnes, OneMax, and BinVal in $O(n \log n)$ function evaluations in expectation and with high probability (Thms. 2 and 3 and Cor. 2), which has not been proven before for any other EDA or classical EA (for further details, see Table 1).

We also observe that the analysis for LeadingOnes can easily be modified to also show an $O(n \log n)$ run time for the *binary value* function BinVal, which is a linear function with exponentially growing coefficients. This result is interesting in that it indicates that the sig-cGA has asymptotically the same run time on BinVal and OneMax. In contrast, for the classic cGA it is known [22] that the run times on OneMax and BinVal differ significantly.

We then show that two previously regarded algorithms which solve LeadingOnes in $O(n \log n)$ time behave poorly on OneMax. The run time of the scGA proposed in [25] is $\Omega(2^{\Theta(\min\{n,1/\rho\})})$ (Thm. 4), where $1/\rho$ is an algorithm-specific parameter controlling the strength of the model update and denotes the hypothetical population size of the algorithm. For the convex search algorithm (CSA) proposed in [35], we prove that the run time, even when adding suitable restart schemes, is asymptotically larger than any polynomial (Thm. 6). These results, together with the large number of existing results, suggest that none of the previously known algorithms performs exceptionally well on both OneMax and LeadingOnes.

These results, the positive ones for the sig-cGA using a longer history of the search process and the negative ones for other algorithms not exploiting a longer history, suggest that a fruitful direction for the future development of the field of evolutionary computation (EC; not restricted to theory) is the search for algorithms that enrich the classic generational approaches with mechanisms that profit from regarding more than one generation. We discuss this in more detail in the conclusions of this paper. We note that, from a practical point of view, our algorithm not only showed a performance not seen so

far with other algorithms, it is also easier to use since, unlike with most other EDAs, the delicate choice of the update strength is obsolete.

This paper extends our previous results on the sig-cGA [15] by proving an upper bound of the sig-cGA on BinVal (Cor. 2) and a lower bound of the CSA on OneMax (Thm. 6).

2 Preliminaries

In this paper, 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 $i \in [n] \coloneqq [1,n] \cap \mathbb{N}$, 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 (OM) and LeadingOnes (LO) defined by

$$\mathrm{OM}(x) = \sum_{i \in [n]} x_i \qquad \text{and} \tag{1}$$

$$\mathrm{LO}(x) = \sum_{i \in [n]} \prod_{j \in [i]} x_j \ . \tag{2}$$

Intuitively, OM returns the number of 1s of an individual, whereas LO returns the longest sequence of consecutive 1s, starting from the left. Note that the all-1s bit string is the unique global optimum for both functions.

In Table 1, we state the asymptotic run times of many algorithms on these two functions. We note that (i) the black-box complexity of OM is $\Theta(n/\log n)$, see [2, 24], and (ii) the black-box complexity of LO 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 these two problems.

For our calculations, we shall regularly use the following well-known variance-based additive Chernoff bounds (see, e.g., the respective Chernoff bound in [8]).

Theorem 1 (Variance-based Additive Chernoff Bounds)**.** *Let $X_1, \dots, X_n$ 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 $m = \min\{\lambda^2/\sigma^2, \lambda\}$,*

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

We say that an event A occurs with high probability (w.h.p.) 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 $k \in [|H|]$, let $H[k]$ denote the *last* k bits in H. In addition to that, $\emptyset$ denotes the empty string.

Algorithm 1: The sig-cGA with parameter ε and significance function sig_ε (eq. (3)) optimizing f

$t \leftarrow 0$;
for $i \in [n]$ **do** $\tau_i^{(t)} \leftarrow \frac{1}{2}$ and $H_i \leftarrow \emptyset$;
repeat
  $x, y \leftarrow$ offspring sampled with respect to $\tau^{(t)}$;
  $x \leftarrow$ winner of x and y with respect to f;
  for $i \in [n]$ **do**
    $H_i \leftarrow H_i \circ x_i$;
    if $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \mathrm{up}$ **then** $\tau_i^{(t+1)} \leftarrow 1 - 1/n$;
    else if $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \mathrm{down}$ **then** $\tau_i^{(t+1)} \leftarrow 1/n$;
    else $\tau_i^{(t+1)} \leftarrow \tau_i^{(t)}$;
    if $\tau_i^{(t+1)} \neq \tau_i^{(t)}$ **then** $H_i \leftarrow \emptyset$;
  $t \leftarrow t + 1$;
until termination criterion met;

3 The Significance-based Compact Genetic Algorithm

Before we present our algorithm sig-cGA in detail in Section 3.1, we provide more information about the *compact genetic algorithm* (cGA [27]), which the sig-cGA as well as the scGA are based on.

The cGA is a univariate EDA, that is, it assumes independence of the bits in the search space. As such, it keeps a vector of probabilities $(\tau_i)_{i\in[n]}$ (the *frequency vector*). In each iteration, two individuals *(offspring)* are sampled in the following way with respect to τ: 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 $j \neq i$.

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 then 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,[4] 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.

[4] 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.

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

Similar to the 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 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 law with $|H_i|$ tries and a success probability of τ_i. We call this our *hypothesis*. 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 s to overcome. For all $\varepsilon, \mu \in \mathbb{R}^+$, where μ is the expected value of our hypothesis and ε is an algorithm-specific parameter:

$$s(\varepsilon, \mu) = \varepsilon \max\left\{ \sqrt{\mu \ln n}, \ln n \right\} .$$

Note that $\sqrt{\mu}$ basically describes the standard deviation of our hypothesis, and the logarithmic factor increases this value such that a deviation does not happen w.h.p. The maximum ensures that we consider at least logarithmically many samples before we conclude that we found a significance, eliminating wrong updates due to small samples sizes w.h.p. The parameter ε effectively turns into the exponent of the w.h.p. bounds. Thus, a larger value of ε decreases the probability of detecting a false significance by a polynomial amount. However, it also increases the number of samples necessary in order to change a frequency. This results in a linear factor of ε in the run time. We provide more details on how ε should be chosen at the end of this subsection (after Cor. 1).

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 $\Pr\big[|X - E[Y]| \leq s(\varepsilon, E[Y])\big] \leq n^{-c}$.

We now state our significance function $\mathrm{sig}_\varepsilon\colon \{\frac{1}{n}, \frac{1}{2}, 1-\frac{1}{n}\} \times \{0,1\}^* \to \{\mathrm{up}, \mathrm{stay}, \mathrm{down}\}$, 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 $p \in \{\frac{1}{n}, \frac{1}{2}, 1-\frac{1}{n}\}$ and all $H \in \{0,1\}^*$, we define, with ε being a parameter of the sig-cGA, recalling that $H[k]$ denotes the last k bits of H,

$$\mathrm{sig}_\varepsilon(p,H) = \begin{cases} \mathrm{up} & \text{if } p \in \{\frac{1}{n}, \frac{1}{2}\} \wedge \exists m \in \mathbb{N}\colon \\ & \quad \|H[2^m]\|_1 \geq 2^m p + s\big(\varepsilon, 2^m p\big), \\ \mathrm{down} & \text{if } p \in \{\frac{1}{2}, 1-\frac{1}{n}\} \wedge \exists m \in \mathbb{N}\colon \\ & \quad \|H[2^m]\|_0 \geq 2^m (1-p) + s\big(\varepsilon, 2^m (1-p)\big), \\ \mathrm{stay} & \text{else.} \end{cases} \tag{3}$$

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 sig-cGA does not detect a significance at a position with no bias in selection (i.e., a *false significance*) w.h.p.

Lemma 1. *Consider the sig-cGA (Alg. 1) with $\varepsilon \geq 1$. Further, consider a position $i \in [n]$ and an iteration such that the distribution X of 1s of H_i follows a binomial law with k tries and success probability τ_i, i.e., there is no bias in selection at position i. Then τ_i changes in this iteration with a probability of 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 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 $k' \leq k$ 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 $\mathrm{Var}[X'] = k'\tau_i(1-\tau_i) \leq k'\tau_i'$. Since we want to apply Theorem 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 $(k'\tau_i' \ln n)^{1/2} \leq \ln n$, which is equivalent to $k' \leq (1/\tau_i') \ln n$. Note that $\lambda^2/\sigma^2 \geq \varepsilon^2 \ln n \geq \ln n$, as $\varepsilon \geq 1$. Thus, $\min\{\lambda^2/\sigma^2, \lambda\} \geq \varepsilon \ln n$.

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

Combining both cases and applying Theorem 1, we get

$$\begin{aligned}\Pr[X' \geq k'\tau_i' + s(\varepsilon, k'\tau_i')] &= \Pr[X' \geq E[X'] + \lambda] \\ &\leq e^{-\frac{1}{3}\min\left\{\frac{\lambda^2}{\sigma^2}, \lambda\right\}} \leq e^{-\frac{\varepsilon}{3}\ln n} = n^{-\frac{\varepsilon}{3}} .\end{aligned}$$

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 1 bounds the probability of detecting a false significance within a single iteration, assuming no bias in selection. The following corollary trivially bounds the probability of detecting a false significance within any number of iterations.

Corollary 1. *Consider the sig-cGA (Alg. 1) with $\varepsilon \geq 1$ 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 at least one frequency changes during an interval of $k' \leq k$ iterations with a probability of at most $k' n^{1-\varepsilon/3} \log_2 k$.*

Proof. For any $i \in [n]$ during any of the k iterations, by Lemma 1, the probability that τ_i changes is at most $n^{-\varepsilon/3} \log_2 k$. Via a union bound over all k' relevant iterations and all n frequencies, the statement follows. □

Intuitively, this corollary states that ε should be chosen such that the term $k' n^{1-\varepsilon/3} \log_2 k$ represents the desired error probability, where k' is the length of an interval such that a frequency only drops with the error probability. Assuming that one chooses $k' = \Theta(n^r)$ for some constant $r > 0$ and desires an error probability of at most n^{-q} for some constant $q > 0$ (ignoring constant factors and the logarithm), it makes sense to choose $\varepsilon \geq 3(r + 1 + q)$.

3.2 Efficient Implementation of the sig-cGA

Recall that, in order to save on the computational cost of checking for a significance, we only do so in historic data in lengths of powers of 2. By precomputing the number of 1s in each such interval, checking a single history for a significance can be done in time logarithmically in its length. Note that the update to this precomputed data can also be done in logarithmic time, as each iteration only a single bit is added to the history and thus the number of 1s can only differ by at most one per interval from one iteration to the next. Consequently, the loop of the sig-cGA has a computational cost of $O(\sum_{i=1}^{n} \log |H_i|)$. Since a history can never be longer than the run time T of the sig-cGA, its total computational cost is $O(nT \log T)$. In comparison, many EAs have

an extra cost of $O(n)$ per iteration. Thus, our significance-based approach is only more costly by a factor of $O(\log T)$.

One drawback of the approach above is that the full history needs to be stored. Thus, we describe a way of condensing a history to a size only logarithmic in the length of the full history. This approach does not allow anymore to access the exact number of 1s (or 0s) in all power-of-two length histories. However, for each $\ell \in [|H_i|]$, it yields the number of 1s in some interval of length ℓ' with $\ell \leq \ell' < 2\ell$. For reasons of readability, we nevertheless regard the original sig-cGA in the subsequent analyses, but it is quite evident that the mildly different accessibility of the history in our condensed implementation does not change our result.

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 starting with the newest element, we check if there are three consecutive blocks of the same size, and if so, we merge the two oldest 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 $j \in [0..k-1]$. 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 $j \in [0..k]$.

For the pseudocode of this approach, assume that a list element has a pointer to the next list element (next), its previous element (prev), and stores an integer value (load).

Algorithm 2: The algorithm used by the sig-cGA in order to condense a history H, given a new bit value x. Let L be a list with elements of non-decreasing load.

```
1  X ← a new list element with load x;
2  Append X to the head of L;
3  C ← head of L;
4  N ← C.next;
5  r ← 0;
6  while N is not null do
7      if C.load = N.load then r ← r + 1;
8      if r = 2 then
9          r ← 0;
10         X ← a new list element with load C.load+ N.load;
11         C.prev.next ← X;
12         X.next ← N.next;
13         N ← X;
14     C ← N;
15     N ← N.next;
```

4 Run Time Results for LO and OM

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 LO and OM. We also note that the optimization process for the binary value function can be analyzed with arguments very similar to those for the LO process. Consequently, we here have an $O(n \log n)$ run time as well. Further, we consider the number of iterations T until the sig-cGA finds the optimal solution. Since it generates two offspring each iteration, the number of fitness function evaluations is at most $2T$.

Note that the sig-cGA treats 1s and 0s symmetrically, that is, it is unbiased in the sense of Lehre and Witt [33]. Hence, all results in this section hold for any type of function as defined in eqs. (1) or (2) where, for any position $i \in [n]$, a bit x_i can be flipped to $1 - x_i$ instead or swapped with another bit x_j.

The following lemma states a useful bound for convex combinations. We use it in order to bound the probability of an event that we decomposed into an event and its complement.

Lemma 2. *Let $\alpha, \beta, x, y \in \mathbb{R}$ such that $x \leq y$ and $\alpha \leq \beta$. Then $\alpha x + (1 - \alpha)y \geq \beta x + (1 - \beta)y$.*

4.1 Analysis of LO

We show that the frequencies are set to $1 - 1/n$ sequentially from the most significant bit position to the least significant, that is, from left to right. w.h.p., no frequency is decreased until the optimization process is finished. Thus, a frequency τ_i will stay at $1/2$ until all of the frequencies to its left are set to $1 - 1/n$. Then τ_i will become relevant for selection, as all of the frequencies left to it will only sample 1s w.h.p. This results in a significant surplus of 1s being saved at position i, and τ_i will be set to $1 - 1/n$ within $O(\log n)$ iterations and remain there. Then frequency τ_{i+1} becomes relevant for selection. As we need to set n frequencies to $1 - 1/n$, we get a run time of $O(n \log n)$.

Theorem 2. *Consider the sig-cGA (Alg. 1) with $\varepsilon > 12$ being a constant. Its run time on* LO *is $O(n \log n)$ w.h.p. and in expectation.*

Proof. We split this proof into two parts and start by showing that the run time is $O(n \log n)$ w.h.p. Then we prove the expected run time.

Run time w.h.p. For the first part of the proof, we consider the first $O(n \log n)$ iterations of the sig-cGA and condition on the event that no frequency decreases during this time, i.e., no (false) significance of 0s is detected. Since, for any position $i \in [n]$ in LO, having a 1 is always at least as good as having a 0, a 1 is saved in H_i with a probability of at least τ_i. Hence, by Corollary 1, no frequency decreases in the first $O(n \log n)$ iterations with a probability of at least $1 - O(n^{2-\varepsilon/3} \log^2 n)$. As $\varepsilon > 12$, for an $\varepsilon' > 2$, this probability is at least $1 - O(n^{-\varepsilon'})$, which is w.h.p. In the following, we condition on this event.

We now prove that the history of the leftmost position with a frequency at $1/2$ saves 1s significantly more often than 0s such that the frequency is set to $1 - 1/n$ after $O(\log n)$

iterations. For the second part of the proof, we use a similar argument, but the frequency is at $1/n$, and it takes $O(n \log n)$ steps to get to $1 - 1/n$. Since the calculations for both scenarios are very similar, we combine them.

Consider a position $i \in [n]$ and any of the first $O(n \log n)$ iterations such that $\tau_i \in \{1/n, 1/2\}$ and, for all positions $j < i$, $\tau_j = 1 - 1/n$. Let O denote the event that we save a 1 in H_i this iteration. We derive an upper bound on the probability to detect the significance of 1s in H_i within $O(\log n)$ iterations by calculating a lower bound on the probability of O. Note that the probability of O is the same for each iteration until τ_i is increased, since we condition on no frequency dropping within the first $O(n \log n)$ iterations.

In order to bound $\Pr[O]$, we consider the event A that the bit at position i of the winning individual is not relevant for selection. That is, A denotes the event that at least one of the two offspring during this iteration has a 0 at a position in $[i-1]$. Thus, if A occurs, a 1 is saved with probability $\tau_i \in \{1/n, 1/2\}$. Otherwise, a 1 is saved if not two 0s are sampled, which has a probability of $1 - (1-\tau_i)^2$. Hence,

$$\Pr[O] = \Pr[A] \cdot \tau_i + \Pr\big[\overline{A}\big] \cdot \big(1 - (1-\tau_i)^2\big) \ ,$$

which is a convex combination of τ_i and $1-(1-\tau_i)^2$. By Lemma 2, decreasing $\Pr\big[\overline{A}\big]$ (as $1 - (1-\tau_i)^2 = \tau_i(2-\tau_i) \geq \tau_i$) results in a lower bound of $\Pr[O]$. Since $\overline{A}$ is equivalent to both offspring having only 1s at positions in $[i-1]$, we see that

$$\Pr\big[\overline{A}\big] = \big(1 - \tfrac{1}{n}\big)^{2(i-1)} \ ,$$

due to our assumption that all frequencies left of position i are at $1 - 1/n$. As this term is minimal for $i = n$, using the well-known inequality $(1-1/n)^{n-1} \geq e^{-1}$, we bound $\Pr\big[\overline{A}\big] \geq e^{-2}$. Further, noting that $1 - (1-\tau_i)^2 \geq (3/2)\tau_i$ for $\tau_i \in \{1/n, 1/2\}$, we bound

$$\begin{aligned}\Pr[O] &\geq (1 - e^{-2}) \cdot \tau_i + e^{-2} \cdot \big(1 - (1-\tau_i)^2\big) \\ &\geq (1 - e^{-2}) \cdot \tau_i + \tfrac{3}{2} e^{-2} \tau_i = \big(1 + \tfrac{1}{2} e^{-2}\big) \cdot \tau_i \ .\end{aligned}$$

Given our bound on the probability of O, we now bound the probability to detect a significance of 1s in H_i within k iterations. To this end, let $X \sim \mathrm{Bin}\big(k, (1 + e^{-2}/2)\tau_i\big)$, and note that X is stochastically dominated by the process of saving 1s at position i. We bound the probability that we do not detect a significance of 1s within k iterations:

$$\begin{aligned}&\Pr[X < k\tau_i + s(\varepsilon, k\tau_i)] \\ &\qquad \leq \Pr\big[X \leq E[X] - \big(\tfrac{k}{2} e^{-2} \tau_i - s(\varepsilon, k\tau_i)\big)\big] \ .\end{aligned}$$

Note that the minuend is positive for $k > (4/\tau_i) e^4 \varepsilon^2 \ln n > \ln n$, which holds due to our assumption $\varepsilon > 12$. Let $c = (4/\tau_i) e^4 \varepsilon^2$, and assume $k \geq 8c \ln n$. Thus, $(k/2) e^{-2} \tau_i - s(\varepsilon, k\tau_i) \geq (k/4) e^{-2} \tau_i =: \lambda$ and $\mathrm{Var}[X] = k(1 + e^{-2}/2)\tau_i\big(1 - (1 + e^{-2}/2)\tau_i\big) \geq \lambda$. By

Theorem 1, noting that $\lambda^2/\mathrm{Var}[X] \leq \lambda$ and using $\mathrm{Var}[X] \leq 2k\tau_i$, we bound

$$\begin{aligned}
&\Pr[X < k\tau_i + s(\varepsilon, k\tau_i)] \leq \Pr\big[X \leq E[X] - \tfrac{k}{4}e^{-2}\tau_i\big] \\
&\leq e^{-\frac{1}{3}\cdot\frac{\lambda^2}{\mathrm{Var}[X]}} \leq e^{-\frac{1}{3}\cdot\frac{k^2e^{-4}\tau_i^2}{16\cdot 2k\tau_i}} = e^{-\frac{1}{3}\cdot\frac{ke^{-4}\tau_i}{32}} \\
&\leq n^{-\frac{1}{3}\cdot\frac{ce^{-4}\tau_i}{4}} = n^{-\frac{\varepsilon^2}{3}} \ .
\end{aligned}$$

Hence, τ_i is set to $1-1/n$ after $(4/\tau_i)e^4\varepsilon^2 \ln n = O\big((1/\tau_i)\log n\big)$ iterations with a probability of at least $1-n^{-\varepsilon^2/3}$. By applying a union bound over all n different possibilities for index i, we see that each frequency (once all frequencies at positions $[i-1]$ are at $1-1/n$) is set to $1-1/n$ within $O\big((1/\tau_i)\log n\big)$ with a probability of at least $1-n^{1-\varepsilon^2/3} \geq 1-n^{-47}$, since $\varepsilon > 12$, which is w.h.p.

Overall, we assume that no frequency decreases during the first $O(n\log n)$ iterations w.h.p., and we showed that each frequency (at $1/2$) is set to $1-1/n$ within $O(\log n)$ iterations w.h.p. once all frequencies to its left are at $1-1/n$. Thus, since there are n frequencies, all frequencies are at $1-1/n$ after $O(n\log n)$ iterations w.h.p. The probability to sample the optimum is now $(1-1/n)^n \geq 1/(2e) = \Omega(1)$. Hence, waiting an additional $O(\log n)$ iterations, the optimum is sampled w.h.p. This concludes the first part of this proof.

Expected run time. Since we showed above that the sig-cGA optimizes LO in $O(n\log n)$ iterations w.h.p., we are left to bound its run time in the event that at least one frequency decreases within the first $O(n\log n)$ iterations. As we discussed at the beginning of the first part of this proof, this only happens with a probability of $O(n^{-\varepsilon'})$, for $\varepsilon' > 2$.

Consider an interval of length t'. By Corollary 1, during the first t iterations, no frequency decreases for t' iterations with a probability of at least $1 - t'n^{1-\varepsilon/3}\log_2 t$. Assume $t \leq n^{2n}$ and $t' = \Theta(n^2\log n)$. Then no frequency decreases for t' iterations w.h.p., since $\varepsilon > 12$.

By using the result calculated in the first part, we see that a leftmost frequency τ_i at $1/n$ is increased during $O\big((1/\tau_i)\log n\big) = O(n\log n)$ iterations w.h.p. Thus, in overall, the sig-cGA finds the optimum during an interval of length $t' = \Theta(n^2\log n)$ w.h.p., as n frequencies need to be increased to $1-1/n$. We pessimistically assume that the optimum is only found with a probability of at least $1/2$ during t' iterations. Hence, the expected run time in this case is $2t' = \Theta(t')$.

Last, we assume that we did not find the optimum during n^{2n} iterations, which only happens with a probability of at most $2^{-n^{2n}/t'}$. Then, the expected run time is at most n^n by pessimistically assuming that all frequencies are at $1/n$.

We conclude the proof by combining all of the three different regimes we just discussed, we see that we can bound the expected run time by

$$O(n\log n) + O(n^{-\varepsilon'}) \cdot O(t') + 2^{-n^{2n}/t'} \cdot n^n = O(n\log n) \ . \qquad \square$$

The proof of Theorem 2 shows that the sig-cGA rapidly makes progress when optimizing LO. In fact, after $O(i\log n)$ iterations, with $i \in [n]$, the sig-cGA finds a solution

with fitness i w.h.p. (if i is large) and in expectation. Thus, in the fixed-budget perspective introduced by Jansen and Zarges [29], the sig-cGA performs very well on LO. For comparison, for the $(1+1)$ EA, it is known that the time to reach a fitness of i is $\Theta(in)$ in expectation and (again, when i is sufficiently large) w.h.p., see [12].

The reason that the sig-cGA optimizes LO so quickly is that the probability of saving a 1 at position i is increased by a constant factor once all frequencies at positions less than i are at $1-1/n$. This boost is a result of position i being the most relevant position for selection, assuming that all bits at positions less than i are 1.

Binary Value A very similar boost in relevance occurs when considering the function BinVal (BV), which returns the bit value of a bit string. Formally, BV is defined as

$$\mathrm{BV}(x) = \sum_{i=1}^{n} 2^{n-i} x_i \ .$$

Note that the most significant bit is the leftmost.

BV imposes a lexicographic order from left to right on a bit string x, since a bit x_i has a greater weight than the sum of all weights at positions greater than i. This is similar to LO. The main difference is that, for BV, a position i can also be relevant for selection when bits at positions less than i are 0. More formally, for LO, position i is only relevant for selection when all of the bits at positions less than i are 1, whereas position i is relevant for selection for BV when all the bits at positions less than i are *the same*. With this insight, we adapt the proof of Theorem 2 for BV.

Corollary 2. *Consider the sig-cGA (Alg. 1) with $\varepsilon > 12$ being a constant. Its run time on* BV *is $O(n \log n)$ w.h.p. and in expectation.*

4.2 Analysis of OM

In order to analyze how likely it is that two individuals sampled from the sig-cGA have the same OM value, we use the following estimate, whose proof can be found, e.g., in [20].

Lemma 3. *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* $\Pr[X = k] = \Omega\left(\frac{1}{\sqrt{\ell}}\right)$.

For the proof of the run time of the sig-cGA on OM, we show that, during *each* of the first $O(n \log n)$ iterations, each position can become relevant for selection with a decent probability of $\Omega(1/\sqrt{n})$. In contrast to LO, there is no sudden change in the probability that 1s are saved. Thus, it takes $O(n \log n)$ iterations to set a frequency to $1 - 1/n$. However, this is done for all frequencies in parallel. Thus, the overall run time remains $O(n \log n)$.

Theorem 3. *Consider the sig-cGA (Alg. 1) with $\varepsilon > 12$ being a constant. Its run time on* OM *is $O(n \log n)$ w.h.p. and in expectation.*

Proof. We first show that the run time holds w.h.p. Then we prove the expected run time.

Run time w.h.p. We consider the first $O(n \log n)$ iterations and condition on the event that no frequency decreases during that time. This can be argued in the same way as at the beginning in the proof of Theorem 2.

We now show that a single frequency (starting at $1/2$) is increased to $1 - 1/n$ within the first $O(n \log n)$ iterations w.h.p. as long as the other frequencies are at $1/2$ or at $1 - 1/n$. Hence, *all* frequencies are increased during that time w.h.p. when applying a union bound.

Similar to the proof of Theorem 2, when proving the expected run time, we use that, if all frequencies start at $1/n$, they are set to $1 - 1/n$ w.h.p. within $O(n^2 \log n)$ iterations in parallel. Thus, we combine both cases in the following argumentation.

Let $s \in \{1/2, 1/n\}$ denote the starting value of a frequency that we consider, and let $\ell \in [n]$ denote the number of frequencies *not* at $1 - 1/n$ during an arbitrary single iteration. Further, let $i \in [n]$ be a position in that iteration such that $\tau_i = s$. We prove that H_i saves 1s more likely by a factor of $1 + \Theta(1/\sqrt{\ell})$ when compared to the hypothesis. This results in τ_i being increased to $1 - 1/n$ within $O\big((\ell/s) \log n\big)$ iterations.

We determine the bias in saving a 1 by making the following observation: ignoring position i, if the absolute difference in the number of 1s of both offspring is greater than one, then bit i is not relevant for determining which offspring is selected. However, if the difference in the number of 1s (except position i) of both offspring is at most 1, having a 1 at position i makes it more likely for an individual to be selected. We now formalize this idea. To this end, let O denote the event that H_i saves a 1, and let A denote the event that the difference of both offspring (except position i) is greater than 1. Note that in the case of A, the probability to save a 1 is τ_i.

We now consider the case of $\overline{A}$, that is, the absolute difference in the number of 1s of both offspring (excluding position i) is at most one. If it is zero, then H_i saves a 1 if none of the offspring has a 0 at position i. Thus, the respective probability is $1 - (1 - \tau_i)^2 = 2\tau_i - \tau_i^2$. In the case of the numbers of 1 differing by exactly one, a 1 is saved if the winner (with respect to all bits but bit i) has a 1 at position i (which it has with a probability of τ_i), or if the winner has a 0 at position i, the loser has a 1, and the loser wins the tie-breaking. The probability of this event is $(1/2)\tau_i(1 - \tau_i) \geq (1/4)\tau_i$. All in all, the probability to save a 1 conditional on $\overline{A}$ is at least $\tau_i + (1/4)\tau_i = (5/4)\tau_i$, since $(5/4)\tau_i \leq 2\tau_i - \tau_i^2$ for $\tau_i \in \{1/2, 1/n\}$.

Taking both cases together, we bound

$$\Pr[O] \geq \Pr[A] \cdot \tau_i + \Pr\big[\overline{A}\big] \cdot \frac{5}{4}\tau_i \ .$$

By Lemma 2, we lower bound this term even further by calculating a lower bound for $\Pr\big[\overline{A}\big]$. We first show that the frequencies at $1-1/n$ and $1/n$ sample the same bits in both offspring with at least a constant probability. For the $n - \ell$ positions with frequencies at $1 - 1/n$, both offspring have a 1 at the respective positions with a probability of $(1-1/n)^{2(n-\ell)} \geq e^{-2}$, since $n - \ell \leq n - 1$. Analogously, for all positions with frequencies at $1/n$ (but τ_i), both offspring have a 0 at position i also with a probability of at least

$(1-1/n)^{2(n-1)} \geq e^{-2}$. Hence, both offspring have the same bits at all positions with frequencies not at $1/2$ with a probability of at least e^{-4}.

We now consider the number of 1s of an offspring at the remaining $\ell' \leq \ell - 1$ (for $\ell \geq 2$) positions (except i) with frequencies at $1/2$. We call this number Y. Note that the expected value of Y is $\ell'/2$. By Lemma 3, for a $k \in [\ell'/2 \pm \sqrt{\ell'/2}]$, the probability that $Y = k$ is $\Omega(1/\sqrt{\ell'})$. Thus, the probability that both offspring have the same number of 1s at the ℓ' positions we consider is $d/\sqrt{\ell'}$, for a constant $d > 0$, since there are $\sqrt{\ell'}$ possible values of k and the probability that both offspring have k bits as 1 is $\Omega(1/\ell')$. Factoring in the probability of all remaining $n - \ell'$ positions to sample the same values in both offspring and for a sufficiently small constant $d' > 0$, we bound

$$\begin{aligned}\Pr[O] &\geq \left(1 - e^{-4}\frac{d}{\sqrt{\ell'}}\right) \cdot \tau_i + e^{-4}\frac{d}{\sqrt{\ell'}} \cdot \frac{5}{4}\tau_i \\ &\geq \left(1 + \frac{d'}{\sqrt{\ell}}\right)\tau_i \ .\end{aligned}$$

Recall that we assumed $\ell \geq 2$ for this bound. For $\ell = 1$, i.e., $\ell' = 0$, we have $n - 1$ positions with frequencies at $1 - 1/n$ or $1/n$. Thus, $\Pr[\overline{A}] \geq e^{-4}$, as we discussed before. Consequently, we bound $\Pr[O] \geq (1 - e^{-2}) \cdot \tau_i + e^{-2} \cdot (5/4)\tau_i \geq (1 + d'/\sqrt{\ell})\tau_i$ if we choose d' sufficiently small. Overall, we use $(1 + d'/\sqrt{\ell})\tau_i$ as a lower bound for $\Pr[O]$.

Analogous to the proof of Theorem 2, we now consider the probability to detect a significance of 1s in H_i within k iterations. To this end, let $X \sim \mathrm{Bin}\big(k, (1 + d'/\sqrt{\ell})\tau_i\big)$ and note that X is stochastically dominated by the process of saving 1s at position i. We bound the probability to not detect a significance of 1s as follows:

$$\begin{aligned}&\Pr[X < k\tau_i + s(\varepsilon, k\tau_i)] \\ &\qquad \leq \Pr\left[X \leq E[X] - \left(\frac{kd'}{\sqrt{\ell}}\tau_i - s(\varepsilon, k\tau_i)\right)\right] \ .\end{aligned}$$

Let $k \geq 4(\varepsilon^2/d'^2)(\ell/\tau_i)\ln n$. Then $(kd'/\sqrt{\ell})\tau_i - s(\varepsilon, k\tau_i) \geq \big(kd'/(2\sqrt{\ell})\big)\tau_i =: \lambda$. Further note that $\mathrm{Var}[X] = k\tau_i(1-\tau_i) \geq \lambda$ if d' is sufficiently small, which implies $\lambda^2/\mathrm{Var}[X] \leq \lambda$. By Theorem 1 with $\mathrm{Var}[X] \leq k\tau_i$, we see that

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

Hence, τ_i is increased to $1 - 1/n$ within $4(\varepsilon^2/d'^2)(\ell/\tau_i)\ln n = O\big((\ell/\tau_i)\log n\big)$ iterations with a probability of at least $1 - n^{-4\varepsilon^2/3}$. By applying a union bound over all n frequencies, each frequency reaches $1 - 1/n$ within $O\big((\ell/\tau_i)\log n\big)$ iterations with a probability of at least $1 - n^{1-4\varepsilon^2/3} \geq 1 - n^{-191}$, as $\varepsilon > 12$, which is w.h.p.

Since we assume that no frequency drops within the first $O(n \log n)$ iterations w.h.p. and since all frequencies start at $1/2$, all of them reach $1 - 1/n$ within that time w.h.p.

Then, the optimum is sampled during a single iteration with a probability of at least $(1-1/n)^n \geq 1/(2e) = \Omega(1)$. Thus, the optimum is sampled after $O(\log n)$ additional iterations w.h.p.

Expected run time. This part follows the same arguments as outlined in the respective part in the proof of Theorem 2. Different from there, assuming that a frequency is at $1/n$, it now takes $O(n^2 \log n)$ iterations to be increased to $1-1/n$ w.h.p., as we proved above. However, since $\varepsilon > 12$, a union bound over all n frequency again results in all frequencies being increased during $O(n^2 \log n)$ iterations w.h.p. The rest remains the same, which concludes this proof. □

OM vs. LO. While the sig-cGA has the same asymptotic run time on LO and OM (w.h.p. and in expectation), the reasons differ. For LO, the frequencies are increased consecutively to $1-1/n$, where each frequency only needs $O(\log n)$ iterations, which is also the asymptotic minimum number of iterations to do so. This speed results from the sudden boost in probability once a position i becomes relevant, that is, its preceding frequencies are all at $1-1/n$ and thus sample only 1s with at least a constant probability. Given that both offspring have only 1s at positions in $[i-1]$, it suffices that position i has at least one 1, which is quite likely. In contrast, the probability that the bias in selection is also detected at positions after i declines exponentially in the distance to i, making the bias negligible. This fact is also exploited by Friedrich et al. [25] in the analysis (and design) of the scGA, which is why it has the same run time on LO.

For OM, the impact of the bias in selection depends on the number ℓ of other frequencies that are not at $1-1/n$. In order for a position i to detect the bias, the number of 1s in these positions has to almost be identical in both offspring, i.e., it can differ by at most one. This then adds a bias of roughly $1/\sqrt{\ell}$ for saving a 1 in H_i. Since ℓ is large for a long time (for example, $\ell = n$ at the beginning), this bias remains small during that period. However, this bias is there constantly for each position. Thus, all frequencies can be optimized in parallel, whereas for LO this is done sequentially.

OM vs. BV. BV is often considered one extremal case of the class of linear functions, as its weights impose a lexicographic order on the bit positions. The other extreme is OM, where all weights are identical and basically no order among the positions exists. Our results show that the sig-cGA optimizes both functions in $O(n \log n)$. It remains an open question whether the sig-cGA is capable of optimizing any linear function in that time, a feat that the $(1+1)$ EA, a classical EA, is known to be capable of [23]. Contrary to that, it was proven for the cGA (an EDA) that it performs worse on BV than on OM [22]. Thus, a uniform performance on the class of linear functions would be a great feat for an EDA.

We would like to note that the result of Droste [22] considered the cGA without frequency borders, that is, the frequencies could reach values of 0. Once this is the case, the algorithm is stuck (as it only samples 0 at this position) and the optimization fails. It is unknown up to date whether the cGA still performs worse on BV when the frequencies are bound to the interval $[1/n, 1-1/n]$. However, the main idea of Droste's proof that frequencies drop very low remains. Thus, if sufficiently many frequencies were to drop to $1/n$, the cGA would still perform badly on BV. Note that this is exactly the problem

that the sig-cGA circumvents with its update rule, resulting in its run time of $O(n \log n)$.

The only other known EDA run time result for BV was recently proven by Lehre and Nguyen [32]. They show that the PBIL optimizes BV with $O(n^2)$ fitness function evaluations in expectation (considering best parameter choices).

5 Run Time Analysis for the scGA

Another variant of the cGA [27] that is able to optimize LO in $O(n \log n)$ w.h.p. is the *stable compact genetic algorithm* (scGA; Alg. 3) introduced by Friedrich et al. [25] with the intent to provide an EDA that optimizes LO in $o(n^2)$. The update procedure of the scGA is very similar to that of the cGA, that is, a frequency at position i is changed by a value of ρ with respect to the difference of the bits at i of the winner and the loser. However, an update toward $1/2$ is stronger by an additive term of a, where $a \in O(\rho)$ is an additional parameter.

Different from many other EDAs, the scGA does not have a margin and explicitly makes use of the frequency values 0 and 1. In fact, the scGA has another parameter $d \in (1/2, 1)$, which indicates a value that is sufficient in order to set a frequency to 1. The value $1 - d$ is used symmetrically in order to set a frequency to 0. Thus, the scGA fixes frequencies once they leave the interval $(1 - d, d)$.

Theorem 4. *Let $\alpha \in (0, 1]$ be a constant. Consider the scGA with $\rho = O(1/\log n)$, $a = \alpha\rho$, and $1/2 < d \leq 5/6$ with $d = \Theta(1)$. Its run time on* OM *is $\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ in expectation and w.h.p. for a constant $c > 0$.*

Before we prove the theorem, we mention two other theorems that we are going to use in the proof. The first bounds the probability of a randomly sampled bit string having $s \in \{0\} \cup [n]$ 1s. We use it in order to bound the probability of both offspring having the same number of 1s. Note that the values $1/6$ and $5/6$ in the lemma are somewhat arbitrary and can be exchanged for any constant in $(0, 1/2)$ and $(1/2, 1)$, respectively.

Lemma 4 ([41])**.** *Let S denote the sum of n independent Poisson trials with probabilities $\tau_1, \dots, \tau_n$ such that, for all $i \in [n]$, $1/6 \leq \tau_i \leq 5/6$. Then, for all $s \in \{0\} \cup [n]$,*

$$\Pr[S = s] = O\left(\frac{1}{\sqrt{n}}\right) .$$

The next theorem provides an upper bound on the probability of a random process stopping after a certain time. We use it in order to show that it is unlikely for a frequency of the scGA when optimizing OM to get to 1 within a certain number of iterations.

Theorem 5 (Negative Drift; [38, 39])**.** *Let $(X_t)_{t \in \mathbb{N}}$ be real-valued random variables describing a stochastic process over some state space, with $X_0 \geq b$. Suppose there exist an interval $[a, b] \subseteq \mathbb{R}$, two constants $\delta, \varepsilon > 0$, and, possibly depending on $\ell := b - a$, a function $r(\ell)$ satisfying $1 \leq r(\ell) = o\big(\ell/\log(\ell)\big)$ such that, for all $t \in \mathbb{N}$, the following two conditions hold:*

Algorithm 3: The scGA [25] with parameters ρ, a, and d optimizing f

$t \leftarrow 0$;
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;

1. $E[X_{t+1} - X_t \mid X_t \wedge a < X_t < b] \geq \varepsilon$ *and,*

2. *for all* $j \in \mathbb{N}$, $\Pr[|X_{t+1} - X_t| \geq j \mid X_t \wedge X_t > a] \leq \frac{r(\ell)}{(1+\delta)^j}$.

Then there is a constant $c > 0$ *such that, for* $T := \min\{t \in \mathbb{N} \mid X_t \leq a\}$, *it holds that*

$$\Pr\left[T \leq 2^{\frac{c\ell}{r(\ell)}}\right] = 2^{-\Omega\left(\frac{\ell}{r(\ell)}\right)} .$$

We now prove our result.

Proof of Theorem 4. We only show that the run time is in $\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ w.h.p. The statement for the expected run time follows by lower-bounding the terms that occur with a probability of $o(1)$ with 0.

We first prove the bound of $\Omega\big(2^{c/\rho}\big)$. We do so by showing that each frequency will stay in the non-empty interval $(1-d, d) \subset [1/6, 5/6]$ w.h.p. Although OM introduces a bias into updating a frequency, it is too tiny in order to compensate the strong drift toward $1/2$ in the update.

We lower-bound the expected time it takes the scGA to optimize OM by upper-bounding the probability it takes a single frequency to leave the interval $(1-d, d)$. Thus, we condition during the entire proof implicitly on the event that all frequencies are in

the interval $(1-d, d)$. Note that, in this scenario, the probability to sample the optimum during an iteration is at most $(5/6)^n$, which is exponentially small, even for a polynomial number of iterations.

Consider an index $i \in [n]$ with $\tau_i \in (1-d, d)$. We only upper-bound the probability it takes τ_i to reach d. Note that the probability of τ_i reaching $1-d$ is at most that large, as OM introduces a bias for 1s into the selection process. Hence, we could argue optimistically for τ_i reaching $1-d$ as we do for τ_i reaching d by swapping 1s for 0s and considering $1-\tau_i$ instead.

Let T denote the first point in time t such that $\tau_i^{(t)} \geq d$. We want to apply Theorem 5 and show that it is unlikely for τ_i to reach d within $2^{c/\rho}$ iterations. Hence, we define the following potential function $g\colon [0,1] \to \mathbb{R}$:

$$g(\tau_i) = \frac{1}{\rho}(1-\tau_i) \ ,$$

which we will use for our frequencies. Note that, at the beginning, τ_i is at $1/2$, i.e., $g(1/2) = 1/(2\rho)$. We stop once $\tau_i \geq d$, i.e., $g(\tau_i) \leq (1-d)/\rho$. Thus, we consider an interval of length $\ell \coloneqq 1/(2\rho) - (1-d)/\rho = (2d-1)/\rho = \Theta(1/\rho)$, as $d > 1/2$ is a constant.

We now argue how an update to τ_i is performed in order to estimate its expected value after an update, which is necessary in order to apply Theorem 5. Consider, similar to the proof of Theorem 3, that the bits of both offspring x and y for all positions but position i have been determined. If the difference of the number of 1s of both offspring is at least 2, i.e., $\|x-y\|_1 - x_i + y_i \geq 2$, then the outcome of neither x_i nor y_i can change the outcome of the selection process. Thus, τ_i increases with probability $\tau_i(1-\tau_i)$, as the winner offspring needs to sample a 1 and the loser a 0. Analogously, in this case, the probability that τ_i decreases is $\tau_i(1-\tau_i)$, too.

If the difference of the number of 1s of both offspring is one, then, in order to increase τ_i, the winner (with respect to all bits but bit i) needs to sample a 1 and the loser a 0, or the winner needs to sample a 0, the loser a 1, and the loser wins. The first case has a probability of $\tau_i(1-\tau_i)$, the second of $(1/2)\tau_i(1-\tau_i)$, due to the uniform selection when the offspring have equal fitness. In order to decrease τ_i, the winner needs to sample a 0, the loser a 1, and the winner has to win, which has a probability of $(1/2)\tau_i(1-\tau_i)$.

If the difference of the number of 1s of both offspring is zero, then τ_i is increased if any offspring samples a 1 and the other samples a 0. This has probability $2\tau_i(1-\tau_i)$. In this case, it is not possible that τ_i is decreased.

In order to estimate the probabilities of when τ_i increases or decreases, we need to estimate the probabilities that the number of 1s of both offspring differ by at least two, differ by exactly one, and differ by exactly zero. Let p_1 denote the probability that this difference is one, and let p_0 denote the probability that the difference is zero. We now bound these probabilities.

Assume that offspring x has k 1s, where $k \in \{0\} \cup [n-1]$, since we assume that bit i has not been sampled yet. For p_0, y needs to sample k 1s as well, and for p_1, y needs to sample $k-1$ or $k+1$ 1s (such that the result is still in $\{0\} \cup [n-1]$). Due to Lemma 4, the probability for y to have this many 1s is $O(1/\sqrt{n})$, as we assume that all frequencies

are in the interval $(1-d, d) \subset [1/6, 5/6]$. Hence, by the law of total probability, we get

$$p_0 = O\Big(\tfrac{1}{\sqrt{n}}\Big) \text{ and } p_1 = O\Big(\tfrac{1}{\sqrt{n}}\Big) .$$

In the following, let $\gamma > 0$ be a constant such that $p_0 \leq \gamma/\sqrt{n}$ and $p_1 \leq \gamma/\sqrt{n}$.

We now consider the drift of $g(\tau_i)$ in any iteration t such that $1/2 < \tau_i^{(t)} < d$, i.e., we show that condition (1) of Theorem 5 holds. Let $\tau = \tau_i^{(t)}$ and $\tau' = \tau_i^{(t+1)}$. Note that conditioning on $g(\tau)$ is the same as conditioning on τ, as g is injective. If τ increases, it changes by ρ, and if it decreases, it changes by $\rho + a$.

$$\begin{aligned}
&E\Big[g(\tau') - g(\tau) \;\Big|\; g(\tau) \wedge \frac{1-d}{\rho} < g(\tau) < \frac{1}{2\rho}\Big] \\
&= \tfrac{1}{\rho} E\big[\tau - \tau' \mid \tau \wedge \tfrac{1}{2} < \tau < d\big] \\
&= \tfrac{1}{\rho}\Big((\rho+a)\big((1-p_0-p_1)\tau(1-\tau) + p_1 \cdot \tfrac{1}{2}\tau(1-\tau)\big) \\
&\qquad - \rho\big((1-p_0-p_1)\tau(1-\tau) + p_1 \cdot \tfrac{3}{2}\tau(1-\tau) \\
&\qquad\qquad + \; p_0 \cdot 2\tau(1-\tau)\big)\Big) \\
&= \tfrac{1}{\rho}\tau(1-\tau)\big((\rho+a)\big(1-p_0-\tfrac{1}{2}p_1\big) - \rho\big(1+p_0+\tfrac{1}{2}p_1\big)\big) \\
&= \tfrac{1}{\rho}\tau(1-\tau)\big(\rho(-2p_0-p_1) + a\big(1-p_0-\tfrac{1}{2}p_1\big)\big) .
\end{aligned}$$

For the negative terms with factor a, by using that $a \leq \rho$ and by applying the bounds on p_0 and p_1, we get

$$\begin{aligned}
&E\Big[g(\tau') - g(\tau) \;\Big|\; g(\tau) \wedge \frac{1-d}{\rho} < g(\tau) < \frac{1}{2\rho}\Big] \\
&\geq \frac{1}{\rho}\tau(1-\tau)\big(a - 3p_0\rho - \tfrac{3}{2}p_1\rho\big) \\
&\geq \frac{1}{\rho}\tau(1-\tau)\Big(a - 5\frac{\gamma}{\sqrt{n}}\rho\Big) .
\end{aligned}$$

Due to $a = \alpha\rho$, there is a sufficiently small constant $\beta > 0$ such that $a - 5\gamma\rho/\sqrt{n} \geq \beta\rho$. Thus, we get

$$\begin{aligned}
E\Big[g(\tau') - g(\tau) \;\Big|\; g(\tau) \wedge \frac{1-d}{\rho} < g(\tau) < \frac{1}{2\rho}\Big] &\geq \beta\tau(1-\tau) \\
&\geq \beta\tfrac{1}{6} \cdot \tfrac{5}{6} ,
\end{aligned}$$

which is constant.

We now show that condition (2) of Theorem 5 holds. For this, we define $r(\ell) = 2$ and $\delta = \sqrt{2}-1 > 0$. Note that $1 \leq r(\ell) = o\big(\ell/\log(\ell)\big) = o\big(1/(\rho\log(1/\rho))\big)$ holds, as $\rho = o(1)$. Since τ can change by at most $\rho + a \leq 2\rho$ during a single update, $g(\tau)$ can change by at most 2. Thus, we only need to bound $\Pr[|g(\tau') - g(\tau)| \geq j \mid g(\tau) \wedge g(\tau) > (1-d)\rho]$

for $j \in \{0,1,2\}$. For all of these three cases, $r(\ell)/\big((1+\delta)^j\big) \geq 1$. Thus, condition (2) trivially holds for all $j \in \mathbb{N}$.

Overall, by applying Theorem 5 and recalling that $\ell = \Theta(1/\rho)$, there are constant $c, c', c'' > 0$ such that

$$\Pr\left[T \leq 2^{\frac{c'\ell}{r(\ell)}}\right] = \Pr\left[T \leq 2^{\frac{c}{\rho}}\right] \leq 2^{-\Omega\left(\frac{1}{\rho}\right)} \leq n^{-c''} .$$

Thus, w.h.p., τ_i does not reach b within $2^{c/\rho}$ iterations, given that all frequencies are in $(1-d, d)$. As discussed before, the probability of τ_i reaching $1-d$ has at most the same probability. Note that conditioning on never sampling the optimum during any of these t iterations increases these probabilities only by a factor of $1 - t(5/6)^n$, which is constant if $t = o(2^{\Theta(n)})$. Otherwise, we choose $2^{\Theta(n)}$ as run time bound. This concludes the proof. □

6 Run Time Analysis for the Convex Search Algorithm

The following *convex search algorithm* was proposed by Moraglio [35]. Its sole parameter is a population size $\mu \in \mathbb{N}$. The algorithm starts with a first population of μ random individuals $x^{(1,1)}, \dots, x^{(1,\mu)} \in \{0,1\}^n$. In each iteration $t = 1, 2, \dots$, the algorithm generates from the current "parent" population $x^{(t,1)}, \dots, x^{(t,\mu)}$ a new "offspring" population $x^{(t+1,1)}, \dots, x^{(t+1,\mu)}$ as follows.

- If the parent population contains only copies of a single individual, the algorithm stops and outputs this solution.
- If all individuals of the parent population have the same fitness, the offspring population is the parent population.
- Otherwise, the individuals with lowest fitness value are removed from the parent population (giving the "reduced parent population") and the offspring population is obtained by μ times independently sampling from the convex hull of the reduced parent population. In other words, for all $i \in [n]$ and $j \in [\mu]$ independently, $x_i^{(t+1,j)}$ is chosen randomly from $\{0,1\}$ if the reduced parent populations contains both an individual having a 0 at the i-th position and an individual having a 1 at this position. If all individuals of the reduced parent population have the same value $b \in \{0,1\}$ in the i-th position, then $x_i^{(t+1,j)} := b$.

The convex search algorithm with $\mu \geq 8\log_2(4n^2+n)$ and a suitable restart strategy was shown to optimize the LO problem in expected time $O(n \log n)$ [35]. We now show that its performance on the OM problem is not very attractive, namely it is asymptotically larger than any polynomial even when employing a suitable restart strategy. We suspect that much stronger lower bounds hold, but given the only moderate general interest in this algorithm so far, we restrict ourselves to this super-polynomial lower bound.

Theorem 6. *Let $c > 0$. Regardless of the population size, a run of the convex search algorithm on* OM *with probability at least $1 - O(n^{-c})$*

- *either reaches a state from which the optimum cannot be found,*
- *or within n^c iterations does not fix any bit-position.*

Consequently, at least $\Omega(n^c)$ iterations are necessary to find the optimum.

While the result may seem natural, proving it is made difficult by the dependencies inflicted from restricting the parent population to all but the lower fitness level. We remove these dependencies by suitable pessimistic estimates (e.g., estimating that a position does not become fixed to 1 when at least Δ zeros are sampled), by suitable domination arguments (cf. [7]), and by first regarding an artificial process in which bits can only be fixed to 1.

Proof of Theorem 6. Since we are aiming at an asymptotic statement, we assume in the following that n is sufficiently large.

To ease the following proof, we first argue that only the case $\mu = \Theta(\log n)$ is interesting. If $\mu \leq \frac{1}{2}\log_2 n$, then with probability

$$1 - (1 - 2^{-\mu})^n \geq 1 - \exp(-2^{-\mu}n) \geq 1 - \exp(-n^{1/2}),$$

at least one of the bit-positions of the initial population is already converged to zero. Let now $\mu \geq K \log_2 n$ for a sufficiently large constant K (which may depend on the constant c). We show that a random population with probability n^{-2c} fixes no bit (that is, the next population is again fully random). By elementary properties of the binomial distribution, with probability at least $1 - (2/3)^\mu$, the lowest fitness value of the random population is below $n/2$. Since the probability of having a fitness of at least $n/2$ is at least $\frac{1}{2}$, the additive Chernoff bound ([8, Theorem 10.7]) gives that with probability at least $1 - \exp(-\mu/8)$, the number μ^+ of individuals having a fitness of at least $n/2$ is at least $\mu/4$.

We now condition on $\mu^+ \geq \mu/4$ and that there is an individual with fitness less than $n/2$ (and recall that this event happens with probability at least $1-(2/3)^\mu-\exp(-\mu/8)$). We first note that in this case the μ^+ individuals with fitness $n/2$ or more surely belong to the reduced population, which defines the next population. For each of these μ^+ individuals and for each of their bit-positions, the probability to have a one is between $1/2$ and $3/4$, since these individuals are random individuals conditional on having at least $n/2$ ones. Since these individuals are stochastically independent, the probability that a bit-position in all these μ^+ individuals has the value 0 is at most $(1/2)^{\mu^+} \leq (1/2)^{\mu/4}$ and the probability is at most $(3/4)^{\mu^+} \leq (3/4)^{\mu/4}$ for the event that they are all one.

In summary, we obtain that the probability that a bit-position becomes fixed is at most $(2/3)^\mu + \exp(-\mu/8) + (1 - (2/3)^\mu - \exp(-\mu/8))n((1/2)^{\mu/4} + (3/4)^{\mu/4})$. By taking the constant K in the lower bound for μ sufficiently large, this probability is at most n^{-2c}. Hence a union bound over n^c iterations shows that within this time frame, with probability $1 - n^{-c}$ no bit-position becomes fixed.

In the remainder we thus assume that $\mu = \Theta(\log n)$ with implicit constants depending on the constant c only.

We first regard the artificial random process which equals a true run of the CSA on OM except that for all bits $i \in [n]$ where the reduced parent population contains only individuals with bit-value 0 we still sample the offspring bits randomly from $\{0,1\}$. In other words, we prevent the algorithm from letting a bit-value converge to the wrong value of 0.

Let $\Delta = \lceil 2 + 2c + 2\frac{\mu}{\log_2 n} \rceil$ and $t = \left\lfloor \frac{2^\mu}{4(2\mu)^\Delta} \right\rfloor$. We call a bit-position $i \in [n]$ at time $s \in [t]$ *unsafe* if the population at time s contains at least $\mu - \Delta$ ones in this bit-position, that is, if $\sum_{j=1}^{\mu} x_i^{(s,j)} \geq \mu - \Delta$, and if this bit-position was determined by sampling random bit-values (that is, not by setting all bit-values to one because the previous reduced population was converged to 1 in this position). Let X_{is} be the indicator random variable for this event.

We easily see that

$$\begin{aligned} \Pr[X_{is} = 1] = \Pr[\mathrm{Bin}(\mu, \tfrac{1}{2}) \geq \mu - \Delta] &\leq 2^{-(\mu-\Delta)} \binom{\mu}{\mu - \Delta} \\ &\leq 2^{-\mu} (2\mu)^\Delta \,, \end{aligned}$$

where the estimate for the binomial distribution is well-known (see [26, Lemma 3] or [8, Lemma 10.37]).

Regarding the correlation of the X_{is}, we see that either X_{is} is a fresh random sample independent from all $X_{i's'}$ with $s' < s$ and $i' \in [n]$ or, namely if the reduced parent population in iteration s has the i-th bit converged, $X_{is} = 0$ with probability one. Consequently, the number $X \coloneqq \sum_{s=1}^{t} \sum_{i=1}^{n} X_{is}$ of unsafe bit-positions in the time frame $[t]$ is dominated by a sum of nt independent Bernoulli random variables with success probability $2^{-\mu}(2\mu)^\Delta$, see [13, Lemma 11] or [8, Lemma 10.22].

For these reasons, we have $E[X] \leq \frac{1}{4}n$ and $\Pr[X \geq \frac{1}{2}n] \leq \exp(-\frac{1}{8}n)$ by the additive Chernoff bound.

We now argue that having at least Δ zeros in some bit-position is often enough sufficient for the position not being fixed to 1. For each $s \in [t]$, let B_s be the following event.

- If in the sampling process of the s-th population at least $\frac{n}{2}$ bit-positions are not already fixed ("fat s-th population"), then B_s is the event that there is a fitness value $z \in [0..n]$ such that at least Δ individuals of the s-th population have fitness exactly z.
- Otherwise ("thin s-th population") let the event B_s be true with probability $p \coloneqq n^{1-\Delta/2}(2\mu)^\Delta$ independent of all other random decisions of the algorithm.

Since a random variable with binomial distribution with parameters n and $\frac{1}{2}$ attains each value in $[0..n]$ with probability at most $2/\sqrt{n}$, this follows from elementary estimates of binomial coefficient, see, e.g., [8, Lemma 4.9], a union bound over the $n+1$ possible

values of z and a similar estimate as above shows

$$\Pr[B_s] \leq (n+1)\binom{\mu}{\Delta} 2^{\Delta} n^{-\Delta/2} \leq n^{1-\Delta/2}(2\mu)^{\Delta} = p$$

also in the first case, where the last estimate exploits that $\Delta \geq 2$.

Denote by $\overline{B} = \bigwedge_{s=1}^{t} \overline{B}_s$ the event that none of the B_s comes true. By a simple union bound,

$$\Pr[(X \leq \tfrac{n}{2}) \wedge \overline{B}] \geq 1 - \exp(-\tfrac{n}{8}) - tp \geq 1 - \exp(-\tfrac{n}{8}) - \tfrac{1}{4}n^{-c}.$$

A simple induction shows that the event "$(X \leq \frac{n}{2}) \wedge \overline{B}$" implies that in each iteration $s \in [2..t]$ at most those bit-positions which have been unsafe before can be converged to one. For $s = 2$ this follows from the fact that all positions of the initial population are sampled randomly; consequently, the event $\overline{B}_2$ means that all fitness values occurred less than Δ times. This, however, implies that only a bit-position i which was unsafe in the first iteration (that is, $X_{i1} = 1$) can be converged in the second population. Since the total number of unsafe positions is at most $n/2$, also the second population is fat, that is, contains at least $n/2$ random bits. Repeating the previous arguments, we obtain that at most bit-positions which where unsafe at least once can be converged to one, and further, that all populations up to time t are fat.

Conditioning on the event "$(X \leq \frac{n}{2}) \wedge \overline{B}$", we now regard the difference between the artificial process and a true run of the CSA. We have just seen that during the run of the artificial process, at least $tn/2$ times a bit-position was sampled randomly (without becoming unsafe). The number of ones in such a bit-position is described by a random variable $(Z \mid Z \leq \mu - \Delta)$, where Z follows a binomal law with parameters μ and $\frac{1}{2}$. In particular, with probability at least $2^{-\mu}$, this number is zero. Note that when a bit-position is sampled with zero ones, then the true process differs from the artificial process and the run of the CSA reaches a state from which it cannot generate the optimum of OM. The probability that none of the at least $tn/2$ safe samplings of variables leads to this negative event is at most

$$\begin{aligned}(1-2^{-\mu})^{tn/2} &\leq \exp\left(-\frac{2^{-\mu}tn}{2}\right) \\ &\leq \exp\left(-\frac{2^{-\mu}n}{2}\frac{2^{\mu}}{2\cdot 4(2\mu)^{\Delta}}\right) = \exp\left(-\frac{n}{16(2\mu)^{\Delta}}\right).\end{aligned}$$

In summary, we see that with probability at least

$$\begin{aligned}\left(1-\exp(-\tfrac{n}{8})-\tfrac{1}{4}n^{-c}\right)\left(1-\exp\left(-\frac{n}{16(2\mu)^{\Delta}}\right)\right) \\ = 1 - O(n^{-c}),\end{aligned}$$

the run of the true CAS fixes a position to zero. □

7 Conclusions

We introduced the novel EDA sig-cGA, which optimizes both OM and LO in $O(n \log n)$ w.h.p. and in expectation. This is the first result of this kind for an EDA or even an EA. These run times are a result of the update process of the sig-cGA: it only updates its probabilistic model if it finds a significance in the history of its samples. In contrast, common EDAs or EAs that are analyzed theoretically do not store the entire history of their samples; EAs keep some samples as their population, and EDAs learn from samples iteratively and store the gained information implicitly in their model.

Since storing the entire history of samples demands a lot of memory if the sig-cGA runs longer, we proposed a method that stores the history compactly while maintaining its important information. We want to note that this method can be improved even further. Currently, the sig-cGA saves new data in each iteration, even if no information is gained. In order to further reduce the memory demands of the sig-cGA, it should save a bit only if it is different from that of the competing offspring, that is, if there actually was a bias in both offspring at that position. Note that this is more similar to how the cGA updates its frequencies. However, if a frequency of the sig-cGA is at $1/2$, the number of samples that can contain important information (that is the pairs $(0, 1)$ and $(1, 0)$) is, in expectation, only half the number of all samples. Thus, the memory is only reduced by roughly a factor of 2.

All in all, the approach of the sig-cGA to reduce run times for a slight increase in memory appears to pay off very well. In this first work, as often in the theory of evolutionary algorithms, we only regarded the two unimodal benchmark functions OM and LO. Since it has been observed, e.g., recently in [16], that insights derived from such analyses can lead to wrong conclusions for more difficult functions, an interesting next step would be to analyze the performance of the sig-cGA on objective functions that have true local optima or that have larger plateaus of equal fitness. Two benchmark functions have been suggested in this context, namely jump functions [23] having an easy to reach local optimum with a scalable basin of attraction and plateau functions [3] having a plateau of scalable diameter around the optimum. We are vaguely optimistic that our sig-cGA has a good performance on these as well. We expect that the sig-cGA, as when optimizing OM, quickly fixes a large number of bits to the correct value and then, different from classic EAs, profits from the fact that the missing bits are sampled with uniform distribution, leading to a much more efficient exploration of the small subhypercube formed by these undecided bits. Needless to say, transforming this speculation into a formal proof would be a significant step forward to understanding the sig-cGA.

From a broader perspective, our work shows that by taking into account a longer history and only updating the model when the history justifies it, the performance of a classic EDA can be improved and its usability can be increased (since the difficult choice of the model update strength is now obsolete). An interesting question from this viewpoint would be to what extent similar ideas can be applied to other well-known EDAs.

From a very broad perspective, our work suggests that generally EC could profit from

enriching the iterative evolutionary process with mechanisms that collect and exploit information over several iterations. So far, such learning-based concepts are rarely used in EC. The only theoretical works in this direction propose a history-based choice of the mutation strength [10] and analyze hyperheuristics that stick to a chosen subheuristic until its performance over the last τ iterations, τ a parameter of the algorithms, appears insufficient (see, e.g., [17] and the references therein).

References

[1] Peyman Afshani, Manindra Agrawal, Benjamin Doerr, Carola Doerr, Kasper Green Larsen, and Kurt Mehlhorn. 2019. The query complexity of finding a hidden permutation. *Discrete Applied Mathematics* (2019), 28–50. https://doi.org/10.1016/j.dam.2019.01.007

[2] Gautham Anil and R. Paul Wiegand. 2009. Black-box search by elimination of fitness functions. In *Proc. of FOGA'09*. ACM, 67–78. https://doi.org/10.1145/1527125.1527135

[3] Denis Antipov and Benjamin Doerr. 2018. Precise runtime analysis for plateaus. In *Proc. of PPSN'18*. Springer, 117–128. https://doi.org/10.1007/978-3-319-99259-4_10

[4] Denis Antipov, Benjamin Doerr, Jiefeng Fang, and Tangi Hetet. 2018. A tight runtime analysis for the $(\mu + \lambda)$ EA. In *Proc. of GECCO'18*. ACM, 1459–1466. https://doi.org/10.1145/3205455.3205627

[5] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. 2014. Unbiased Black-Box Complexity of Parallel Search. In *Proc. of PPSN'14*. Springer, 892–901. https://doi.org/10.1007/978-3-319-10762-2_88

[6] Duc-Cuong Dang, Per Kristian Lehre, and Phan Trung Hai Nguyen. 2019. Level-Based Analysis of the Univariate Marginal Distribution Algorithm. *Algorithmica* 81, 2 (2019), 668–702. https://doi.org/10.1007/s00453-018-0507-5

[7] Benjamin Doerr. 2019. Analyzing randomized search heuristics via stochastic domination. *Theoretical Computer Science* 773 (2019), 115–137. https://doi.org/10.1016/j.tcs.2018.09.024

[8] Benjamin Doerr. 2020. Probabilistic Tools for the Analysis of Randomized Optimization Heuristics. In [18]. 1–87. Also available at https://arxiv.org/abs/1801.06733.

[9] Benjamin Doerr and Carola Doerr. 2018. Optimal Static and Self-Adjusting Parameter Choices for the $(1+(\lambda, \lambda))$ Genetic Algorithm. *Algorithmica* 80, 5 (2018), 1658–1709.

[10] Benjamin Doerr, Carola Doerr, and Jing Yang. 2016. k-bit mutation with self-adjusting k outperforms standard bit mutation. In *Proc. of PPSN'16*. Springer, 824–834. https://doi.org/10.1007/978-3-319-45823-6_77

[11] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. 2019. The $(1+\lambda)$ Evolutionary Algorithm with Self-Adjusting Mutation Rate. *Algorithmica* 81, 2 (2019), 593–631. https://doi.org/10.1007/s00453-018-0502-x

[12] Benjamin Doerr, Thomas Jansen, Carsten Witt, and Christine Zarges. 2013. A method to derive fixed budget results from expected optimisation times. In *Proc. of GECCO'13*. ACM, 1581–1588. https://doi.org/10.1145/2463372.2463565

[13] Benjamin Doerr and Daniel Johannsen. 2010. Edge-based representation beats vertex-based representation in shortest path problems. In *Proc. of GECCO'10*. ACM, 759–766. https://doi.org/10.1145/1830483.1830618

[14] Benjamin Doerr and Marvin Künnemann. 2015. Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm – different asymptotic runtimes for different instances. *Theoretical Computer Science* 561 (2015), 3–23. https://doi.org/10.1016/j.tcs.2014.03.015

[15] Benjamin Doerr and Martin S. Krejca. 2018. Significance-based estimation-of-distribution algorithms. In *Proc. of GECCO'18*. ACM, 1483–1490. https://doi.org/10.1145/3205455.3205553

[16] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast genetic algorithms. In *Proc. of GECCO'17*. ACM, 777–784. https://doi.org/10.1145/3205455.3205563

[17] Benjamin Doerr, Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. 2018. On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proc. of GECCO'18*. ACM, 1015–1022. https://doi.org/10.1145/3205455.3205611

[18] Benjamin Doerr and Frank Neumann. 2020. *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Springer. https://doi.org/10.1007/978-3-030-29414-4

[19] Benjamin Doerr, Frank Neumann, Dirk Sudholt, and Carsten Witt. 2011. Runtime analysis of the 1-ANT ant colony optimizer. *Theoretical Computer Science* 412, 17 (2011), 1629–1644.

[20] Benjamin Doerr and Carola Winzen. 2014. Ranking-based black-box complexity. *Algorithmica* 68 (2014), 571–609. https://doi.org/10.1007/s00453-012-9684-9

[21] Benjamin Doerr, Carsten Witt, and Jing Yang. 2018. Runtime analysis for self-adaptive mutation rates. In *Proc. of GECCO'18*. ACM, 1475–1482. https://doi.org/10.1145/3205455.3205569

[22] Stefan Droste. 2006. A rigorous analysis of the compact genetic algorithm for linear functions. *Natural Computing* 5, 3 (2006), 257–283. https://doi.org/10.1007/s11047-006-9001-0

[23] 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. https://doi.org/10.1016/S0304-3975(01)00182-7

[24] 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. https://doi.org/10.1007/s00224-004-1177-z

[25] Tobias Friedrich, Timo Kötzing, and Martin S. Krejca. 2016. EDAs cannot be balanced and stable. In *Proc. of GECCO'16*. ACM, 1139–1146. https://doi.org/10.1145/2908812.2908895

[26] Christian Gießen and Carsten Witt. 2017. The interplay of population size and mutation probability in the $(1 + \lambda)$ EA on OneMax. *Algorithmica* 78 (2017), 587–609. https://doi.org/10.1007/s00453-016-0214-z

[27] 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.

[28] 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. https://doi.org/10.1162/106365605774666921

[29] Thomas Jansen and Christine Zarges. 2014. Performance analysis of randomised search heuristics operating with a fixed budget. *Theoretical Computer Science* 545 (2014), 39–58. https://doi.org/10.1016/j.tcs.2013.06.007

[30] 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. https://doi.org/10.1145/3040718.3040724

[31] Martin S. Krejca and Carsten Witt. 2020. Theory of Estimation-of-Distribution Algorithms. In [18]. 405–442. Also available at http://arxiv.org/abs/1806.05392.

[32] Per Kristian Lehre and Phan Trung Hai Nguyen. 2018. Level-based analysis of the population-based incremental learning algorithm. In *Proc. of PPSN'18*. Springer, 105–116. https://doi.org/10.1007/978-3-319-99259-4_9

[33] Per Kristian Lehre and Carsten Witt. 2012. Black-box search by unbiased variation. *Algorithmica* 64, 4 (2012), 623–642. https://doi.org/10.1007/s00453-012-9616-8

[34] Johannes Lengler, Dirk Sudholt, and Carsten Witt. 2018. Medium step sizes are harmful for the compact genetic algorithm. In *Proc. of GECCO'18*. ACM, 1499–1506. https://doi.org/10.1145/3205455.3205576

[35] Alberto Moraglio and Dirk Sudholt. 2017. Principled design and runtime analysis of abstract convex evolutionary search. *Evolutionary Computation* 25, 2 (2017), 205–236. https://doi.org/10.1162/EVCO_a_00169

[36] Frank Neumann, Dirk Sudholt, and Carsten Witt. 2009. Analysis of different MMAS ACO algorithms on unimodal functions and plateaus. *Swarm Intelligence* 3, 1 (2009), 35–68. https://doi.org/10.1007/s11721-008-0023-3

[37] Frank Neumann and Carsten Witt. 2009. Runtime analysis of a simple ant colony optimization algorithm. *Algorithmica* 54, 2 (2009), 243–255. https://doi.org/10.1007/s00453-007-9134-2

[38] Pietro S. Oliveto and Carsten Witt. 2011. Simplified drift analysis for proving lower bounds in evolutionary computation. *Algorithmica* 59, 3 (2011), 369–386. https://doi.org/10.1007/s00453-010-9387-z

[39] 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

[40] Martin Pelikan, Mark Hauschild, and Fernando G. Lobo. 2015. Estimation of distribution algorithms. In *Springer Handbook of Computational Intelligence*. 899–928. https://doi.org/10.1007/978-3-662-43505-2_45

[41] Dirk Sudholt and Carsten Witt. 2019. On the Choice of the Update Strength in Estimation-of-Distribution Algorithms and Ant Colony Optimization. *Algorithmica* 81, 4 (2019), 1450–1489. https://doi.org/10.1007/s00453-018-0480-z

[42] Carsten Witt. 2006. Runtime analysis of the (μ + 1) EA on simple pseudo-Boolean functions. *Evolutionary Computation* 14 (2006), 65–86. https://doi.org/10.1162/evco.2006.14.1.65

[43] Carsten Witt. 2019. Upper Bounds on the Running Time of the Univariate Marginal Distribution Algorithm on OneMax. *Algorithmica* 81, 2 (2019), 632–667. https://doi.org/10.1007/s00453-018-0463-0

[44] Weijie Zheng, Guangwen Yang, and Benjamin Doerr. 2018. Working principles of binary differential evolution. In *Proc. of GECCO'18*. ACM, 1103–1110. https://doi.org/10.1145/3205455.3205623