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 cGA (sig-cGA) optimizes the commonly regarded benchmark functions OneMax (OM), LeadingOnes, and BinVal all in quasilinear time, a result shown for no other EDA or evolutionary algorithm so far. For the recently proposed stable compact genetic algorithm—an EDA that tries to prevent erratic model updates by imposing a bias to the uniformly distributed model—we prove that it optimizes OM only in a time exponential in its hypothetical population size. Similarly, we show that the convex search algorithm cannot optimize OM in polynomial time.**

***Index Terms*—Estimation-of-distribution algorithm (EDA), run time analysis, theory.**

I. Introduction

Estimation-of-distribution algorithms (EDAs; [2]) 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.

Manuscript received October 10, 2018; revised April 6, 2019 and September 24, 2019; accepted November 25, 2019. Date of publication November 28, 2019; date of current version December 1, 2020. This work was supported in part by 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, and in part by the COST Action under Grant CA15140. This article extends our results published at GECCO'18 [1]. *(Corresponding author: Martin S. Krejca.)*

B. Doerr is with the Laboratoire d'Informatique, CNRS, École Polytechnique, Institute Polytechnique de Paris, 91120 Palaiseau, France.

M. S. Krejca is with the Hasso Plattner Institute, University of Potsdam, 14482 Potsdam, Germany (e-mail: martin@krejca.de).

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

Digital Object Identifier 10.1109/TEVC.2019.2956633

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.

The theoretical analysis of EDAs has recently gained momentum (see the survey [3]) and has clearly shown that this tradeoff between convergence speed and accumulation of erratic updates can be delicate and nontrivial to understand. Among the most relevant works, Sudholt and Witt [4] and Krejca and Witt [5] proved lower bounds of the expected run times of three common EDAs on the benchmark function OneMax (OM). 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 [4] proved a lower bound of $\Omega(K\sqrt{n} + n\log n)$ for the 2-MMAS$_{ib}$ and the compact genetic algorithm (cGA), where $1/K$ is the step size of the algorithm, and Krejca and Witt [5] proved 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.* [6] that the run time of the cGA on OM 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 [4], 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.* [7] also discussed 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.* [7] considered 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.* [7] then proved 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.* [7] proposed an EDA [called stable compact genetic algorithm (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 OM (Theorem 4). We note that a similar bias toward the middle frequency of $1/2$ was proven for a binary differential evolution algorithm by Zheng *et al.* [8]. Similar to the situation of the scGA, their run time results (partially relying on mean-field assumptions) indicate that LeadingOnes (LO) 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 OM, the results are less conclusive, but they indicate a run time exponential in the population size can occur.

The results of Sudholt and Witt [4], Krejca and Witt [5], and Lengler *et al.* [6], and Friedrich *et al.* [7] drawed 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 article, 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 cGA (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 LO, OM, and BinVal (BV) in $O(n \log n)$ function evaluations in expectation and with high probability (w.h.p.) (Theorems 2 and 3 and Corollary 2), which has not been proven before for any other EDA or classical EA (for further details, see Table I).

We also observe that the analysis for LO can easily be modified to also show an $O(n \log n)$ run time for the *binary value* function BV, 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 BV and OM. In contrast, for the classic cGA it is known [25] that the run times on OM and BV differ significantly.

We then show that two previously regarded algorithms which solve LO in $O(n \log n)$ time behave poorly on OM. The run time of the scGA proposed in [7] is $\Omega(2^{\Theta(\min\{n, 1/\rho\})})$ (Theorem 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 [18], we prove that the run time, even when adding suitable restart schemes, is asymptotically larger than any polynomial (Theorem 5). These results, together with the large number of existing results, suggest that none of the previously known algorithms performs exceptionally well on both OM and LO.

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 article. 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 article extends our previous results on the sig-cGA [1] by proving an upper bound of the sig-cGA on BV (Corollary 2) and a lower bound of the CSA on OM (Theorem 5). Due to space constraints, some proofs can be found in the supplementary material.

II. Preliminaries

In this article, we consider the maximization of pseudo-Boolean functions $f : \{0, 1\}^n \to \mathbb{R}$, where n is a positive integer (fixed for the remainder of this article). 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 OM and LO defined by

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

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

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.

TABLE I

EXPECTED RUN TIMES (NUMBER OF FITNESS EVALUATIONS) OF VARIOUS ALGORITHMS UNTIL THEY FIRST FIND AN OPTIMUM FOR THE TWO FUNCTIONS OM (1) AND LO (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 $(1 + (\lambda, \lambda))$ 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	$\Theta(n \log n)$ [9]	none	$\Theta(n^2)$ [9]	none
$(\mu+1)$ EA	$\Theta(\mu n + n \log n)$ [10]	$\mu = O\big(\mathrm{poly}(n)\big)$	$\Theta(\mu n \log n + n^2)$ [10]	$\mu = O\big(\mathrm{poly}(n)\big)$
$(1+\lambda)$ EA	$\Theta\Big(n \log n + \frac{\lambda n \log \log \lambda}{\log \lambda}\Big)$ [11], [12][a]	$\lambda = O(n^{1-\varepsilon})$	$\Theta(n^2 + \lambda n)$ [11]	$\lambda = O\big(\mathrm{poly}(n)\big)$
$(\mu+\lambda)$ EA	$\Theta\Big(\frac{n \log n}{\lambda} + \frac{n}{\lambda/\mu} + \frac{n \log^+ \log^+ \lambda/\mu}{\log^+ \lambda/\mu}\Big)$ [16]	$\log^+ x := \max\{1, \log x\}$	$\Omega\Big(n^2 + \frac{\lambda n}{\log(\lambda/n)}\Big)$ [13]	–
$\big(1+(\lambda,\lambda)\big)$ GA	$\Theta\Big(\max\Big\{\frac{n \log n}{\lambda}, \frac{n\lambda \log \log \lambda}{\log \lambda}\Big\}\Big)$ [17]	$p = \frac{\lambda}{n}$, $c = \frac{1}{\lambda}$	unknown	–
CSA	$\Omega(n^c)$ [Thm. 10]	$c > 0$	$O(n \log n)$ [18]	$\mu \geq 8 \ln\big((4n+6)n\big)$, restarts
UMDA/PBIL[b]	$\Omega(\lambda\sqrt{n} + n \log n)$ [5]	$\mu = \Theta(\lambda)$	$O(n\lambda \log \lambda + n^2)$ [19], [20]	$\lambda = \Omega(\log n)$, $\mu = \Theta(\lambda)$
	$O(\lambda n)$ [21], [19]	$\mu = \Omega(\log n) \cap O(\sqrt{n})$, $\lambda = \Omega(\mu)$ or $\mu = \Omega(\sqrt{n} \log n)$, $\mu = \Theta(\lambda)$ or $\mu = \Omega(\log n) \cap o(n)$, $\mu = \Theta(\lambda)$		
cGA/2-MMAS$_{\text{ib}}$	$\Omega\Big(\frac{\sqrt{n}}{\rho} + n \log n\Big)$ [4]	$\frac{1}{\rho} = O\big(\mathrm{poly}(n)\big)$	unknown	–
	$O\Big(\frac{\sqrt{n}}{\rho}\Big)$ [4]	$\frac{1}{\rho} = \Omega(\sqrt{n} \log n) \cap O\big(\mathrm{poly}(n)\big)$		
1-ANT	$O(n^2)$ [22][c]	$\rho = \Omega(n^{-1+\varepsilon})$	$O(n^2 \cdot (6e)^{1/(n\rho)})$ [23]	none
			$2^{\Omega(\min\{n, 1/(n\rho)\})}$ [23]	none
MMAS*	$O\big(\frac{n \log n}{\rho}\big)$ [24]	$\rho = O(1)$	$O\big(n^2 + \frac{n \log n}{\rho}\big)$ [24]	$\rho = O(1)$
			$\Omega\big(n^2 + \frac{n}{-\rho \log(2\rho)}\big)$ [24]	$\rho = 1/\mathrm{poly}(n)$
scGA	$\Omega\big(\min\{2^{\Theta(n)}, 2^{c/\rho}\}\big)$ [Thm. 9]	$1/\rho = \Omega(\log n)$, $a = \Theta(\rho)$, $d = \Theta(1)$, $c > 0$	$O(n \log n)$ [7]	$1/\rho = \Theta(\log n)$, $a = O(\rho)$, $d = \Theta(1)$
sig-cGA (Alg. 1)	$O(n \log n)$ [Thm. 8]	$\varepsilon > 12$	$O(n \log n)$ [Thm. 5]	$\varepsilon > 12$

[a]Better run time bounds for the $(1+\lambda)$ EA are known if the mutation rate is (i) fitness-dependent [13], (ii) self-adjusting [14], or (iii) self-adaptive [15].
[b]The results shown for PBIL are the results of UMDA if not mentioned otherwise, since the latter is a special case of the former.
[c]For $\rho \geq (n-2)/(3n-2)$, the algorithm simulates the $(1+1)$ EA and has a run time of $\Theta(n \log n)$.

In Table I, we state the asymptotic run times of many algorithms on these two functions. We note that 1) the black-box complexity of OM is $\Theta(n/\log n)$, see [26], [27] and 2) the black-box complexity of LO is $\Theta(n \log \log n)$, see [28], 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 the respective Chernoff bound in [29]).

Theorem 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 $m = \min\{\lambda^2/\sigma^2, \lambda\}$

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

We say that an event A occurs w.h.p. if there is a $c = \Omega(1)$ such that $\Pr[A] \geq (1 - n^{-c})$.

Lastly, 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.

III. SIGNIFICANCE-BASED COMPACT GENETIC ALGORITHM

Before we present our algorithm sig-cGA in detail in Section III-A, we provide more information about the (cGA [30]), 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

Algorithm 1: sig-cGA With Parameter ε and Significance Function sig_ε (3) 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 $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \mathrm{up}$ then $\tau_i^{(t+1)} \leftarrow 1 - 1/n$;
9          else if $\mathrm{sig}_\varepsilon(\tau_i^{(t)}, H_i) = \mathrm{down}$ 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;
```

frequency τ_i getting stuck at 0 or 1,[1] 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 1 s.

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.

A. Detailed Description of the sig-cGA

Similar to the cGA, our new algorithm—the (sig-cGA; Algorithm 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) 1/2: starting value;
2) 1/n: significance for 0s was detected;
3) 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 section (after Corollary 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 $\Pr[|X - E[Y]| \leq s(\varepsilon, E[Y])] \leq n^{-c}$.

We now state our significance function $\mathrm{sig}_\varepsilon : \{(1/n), (1/2), 1 - (1/n)\} \times \{0,1\}^* \to \{\mathrm{up}, \mathrm{stay}, \mathrm{down}\}$, 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 $p \in \{(1/n), (1/2), 1 - (1/n)\}$ 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

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

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 (Algorithm 1) with $\varepsilon \geq 1$. Further, consider a position $i \in [n]$ and an iteration such that the distribution X of 1 s 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 $n^{-\varepsilon/3} \log_2 k$.

Lemma 1 bounds the probability of detecting a false significance within a single iteration, assuming no bias in selection.

[1]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.

The following corollary trivially bounds the probability of detecting a false significance within any number of iterations.

Corollary 1: Consider the sig-cGA (Algorithm 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 $k'n^{1-\varepsilon/3}\log_2 k$.

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 $\varepsilon \geq 3(r + 1 + q)$.

B. 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 1 s 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 1 s 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. In the supplementary material, 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 1 s (or 0s) in all power-of-two length histories. However, for each $\ell \in [|H_i|]$, it yields the number of 1 s 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.

IV. 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 [31]. Hence, all results in this section hold for any type of function as defined in (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 $\alpha x + (1-\alpha)y \geq \beta x + (1-\beta)y$.

A. 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 1 s w.h.p. This results in a significant surplus of 1 s 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 (Algorithm 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 1 s 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 1 s 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 $1-\tau_i^2$. Hence

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

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

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

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 $\Pr[\overline{A}] \geq e^{-2}$. Further, noting that $1-\tau_i^2 \geq (3/2)\tau_i$ for $\tau_i \in \{1/n, 1/2\}$, we bound

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

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

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

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 $k \geq 4c\ln n$. Thus, $(k/2)e^{-2}\tau_i - s(\varepsilon, k\tau_i) \geq (k/4)e^{-2}\tau_i =: \lambda$ and $\mathrm{Var}[X] = k\tau_i(1-\tau_i) \geq \lambda$. By Theorem 1, noting that $\lambda^2/\mathrm{Var}[X] \leq \lambda$ and using $\mathrm{Var}[X] \leq k\tau_i$, we bound

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

Hence, τ_i is set to $1-1/n$ after $(4/\tau_i)e^4\varepsilon^2\ln n = O((1/\tau_i)\log n)$ 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 $O((1/\tau_i)\log n)$ 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 $O((1/\tau_i)\log n) = O(n\log n)$ 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 $2t' = \Theta(t')$.

Lastly, 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

$$O(n\log n) + O\left(n^{-\varepsilon'}\right)\cdot O\left(t'\right) + 2^{-n^{2n}/t'}\cdot n^n = O(n\log n). \quad\blacksquare$$

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 [32], 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 [33].

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 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 (Algorithm 1) with $\varepsilon > 12$ being a constant. Its run time on BV is $O(n \log n)$ w.h.p. and in expectation.

B. 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 [34].

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 $\Pr[X = k] = \Omega(1/\sqrt{\ell})$.

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 1 s 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 (Algorithm 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 1 s 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 $O((\ell/s) \log n)$ 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 1 s 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 1 s (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 1 s 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

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

By Lemma 2, we lower bound this term even further by calculating a lower bound for $\Pr[\overline{A}]$. 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 1 s 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 1 s 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 1 s in H_i within k iterations. To this end, let $X \sim \mathrm{Bin}(k, (1 + d'/\sqrt{\ell})\tau_i)$ and note that X is stochastically dominated by the process of saving 1 s at position i. We bound the probability to not detect a significance of 1 s as follows:

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

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

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

Hence, τ_i is increased to $1 - 1/n$ within $4(\varepsilon^2/d'^2)(\ell/\tau_i)\ln n = O((\ell/\tau_i)\log n)$ 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 $O((\ell/\tau_i)\log n)$ iterations with a probability of at least $1 - n^{1-4\varepsilon^2/3} \ge 1 - n^{-191}$, 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 $(1 - 1/n)^n \ge 1/(2e) = \Omega(1)$. 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 Versus 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 1 s 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.* [7] 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 1 s 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 Versus 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 [9]. Contrary to that, it was proven for the cGA (an EDA) that it performs worse on BV than on OM [25]. 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 [25] 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 [20]. They show that the PBIL optimizes BV with $O(n^2)$ fitness function evaluations in expectation (considering best parameter choices).

V. Run Time Analysis for the scGA

Another variant of the cGA [30] that is able to optimize LO in $O(n\log n)$ w.h.p. is the (scGA; found in the supplementary material) introduced by Friedrich *et al.* [7] 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 $1/2 < d \le 5/6$ with $d = \Theta(1)$. Its run time on OM is $\Omega(\min\{2^{\Theta(n)}, 2^{c/\rho}\})$ in expectation and w.h.p. for a constant $c > 0$.

VI. Run Time Analysis for the Convex Search Algorithm

The following CSA was proposed by Moraglio and Sudholt [18]. 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.

1) If the parent population contains only copies of a single individual, the algorithm stops and outputs this solution.
2) If all individuals of the parent population have the same fitness, the offspring population is the parent population.
3) 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 ith 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 ith position, then $x_i^{(t+1,j)} := b$.

The CSA with $\mu \geq 8\log_2(4n^2 + n)$ and a suitable restart strategy was shown to optimize the LO problem in expected time $O(n \log n)$ [18]. 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 5: Let $c > 0$. Regardless of the population size, a run of the CSA on OM with probability at least $1 - O(n^{-c})$:

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

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

VII. Conclusion

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 EAs, we only regarded the two unimodal benchmark functions OM and LO. Since it has been observed, e.g., recently in [35], 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 [9] having an easy to reach local optimum with a scalable basin of attraction and plateau functions [36] 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, this article 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, this article 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 [37] 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 [38] and the references therein).

References

[1] B. Doerr and M. S. Krejca, "Significance-based estimation-of-distribution algorithms," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, 2018, pp. 1483–1490.
[2] M. Pelikan, M. Hauschild, and F. G. Lobo, "Estimation of distribution algorithms," in *Springer Handbook of Computational Intelligence*. Heidelberg, Germany: Springer, 2015, pp. 899–928.
[3] M. S. Krejca and C. Witt, *Theory of Estimation-of-Distribution Algorithms*. Cham, Switzerland: Springer, 2020, pp. 405–442. [Online]. Available: http://arxiv.org/abs/1806.05392

[4] D. Sudholt and C. Witt, "On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization," *Algorithmica*, vol. 81, no. 4, pp. 1450–1489, 2019.

[5] M. S. Krejca and C. Witt, "Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax," in *Proc. 14th ACM/SIGEVO Conf. Found. Genet. Algorithms (FOGA)*, 2017, pp. 65–79.

[6] J. Lengler, D. Sudholt, and C. Witt, "Medium step sizes are harmful for the compact genetic algorithm," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*. 2018, pp. 1499–1506.

[7] T. Friedrich, T. Kötzing, and M. S. Krejca, "EDAs cannot be balanced and stable," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Denver, CO, USA, 2016, pp. 1139–1146.

[8] W. Zheng, G. Yang, and B. Doerr, "Working principles of binary differential evolution," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Kyoto, Japan, 2018, pp. 1103–1110.

[9] S. Droste, T. Jansen, and I. Wegener, "On the analysis of the (1+1) evolutionary algorithm," *Theor. Comput. Sci.*, vol. 276, nos. 1–2, pp. 51–81, 2002.

[10] C. Witt, "Runtime analysis of the $(\mu+1)$ EA on simple pseudo-Boolean functions," *Evol. Comput.*, vol. 14, no. 1, pp. 65–86, 2006.

[11] T. Jansen, K. A. De Jong, and I. Wegener, "On the choice of the offspring population size in evolutionary algorithms," *Evol. Comput.*, vol. 13, no. 4, pp. 413–440, 2005.

[12] B. Doerr and M. Künnemann, "Optimizing linear functions with the $(1+\lambda)$ evolutionary algorithm—Different asymptotic runtimes for different instances," *Theor. Comput. Sci.*, vol. 561, pp. 3–23, Jan. 2015.

[13] G. Badkobeh, P. K. Lehre, and D. Sudholt, "Unbiased black-box complexity of parallel search," in *Proc. Int. Conf. Parallel Problem Solving Nat. (PPSN)*, 2014, pp. 892–901.

[14] B. Doerr, C. Gießen, C. Witt, and J. Yang, "The $(1+\lambda)$ evolutionary algorithm with self-adjusting mutation rate," *Algorithmica*, vol. 81, no. 2, pp. 593–631, 2019.

[15] B. Doerr, C. Witt, and J. Yang, "Runtime analysis for self-adaptive mutation rates," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Kyoto, Japan, 2018, pp. 1475–1482.

[16] D. Antipov, B. Doerr, J. Fang, and T. Hetet, "A tight runtime analysis for the $(\mu+\lambda)$ EA," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Kyoto, Japan, 2018, pp. 1459–1466.

[17] B. Doerr and C. Doerr, "Optimal static and self-adjusting parameter choices for the $(1+(\lambda,\lambda))$ genetic algorithm," *Algorithmica*, vol. 80, no. 5, pp. 1658–1709, 2018.

[18] A. Moraglio and D. Sudholt, "Principled design and runtime analysis of abstract convex evolutionary search," *Evol. Comput.*, vol. 25, no. 2, pp. 205–236, 2017.

[19] D.-C. Dang, P. K. Lehre, and P. T. H. Nguyen, "Level-based analysis of the univariate marginal distribution algorithm," *Algorithmica*, vol. 81, no. 2, pp. 668–702, 2019.

[20] P. K. Lehre and P. T. H. Nguyen, "Level-based analysis of the population-based incremental learning algorithm," in *Proc. Int. Conf. Parallel Problem Solving Nat. (PPSN)*, 2018, pp. 105–116.

[21] C. Witt, "Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax," *Algorithmica*, vol. 81, no. 2, pp. 632–667, 2019.

[22] F. Neumann and C. Witt, "Runtime analysis of a simple ant colony optimization algorithm," *Algorithmica*, vol. 54, no. 2, pp. 243–255, 2009.

[23] B. Doerr, F. Neumann, D. Sudholt, and C. Witt, "Runtime analysis of the 1-ANT ant colony optimizer," *Theor. Comput. Sci.*, vol. 412, no. 17, pp. 1629–1644, 2011.

[24] F. Neumann, D. Sudholt, and C. Witt, "Analysis of different MMAS ACO algorithms on unimodal functions and plateaus," *Swarm Intell.*, vol. 3, no. 1, pp. 35–68, 2009.

[25] S. Droste, "A rigorous analysis of the compact genetic algorithm for linear functions," *Nat. Comput.*, vol. 5, no. 3, pp. 257–283, 2006.

[26] S. Droste, T. Jansen, and I. Wegener, "Upper and lower bounds for randomized search heuristics in black-box optimization," *Theory Comput. Syst.*, vol. 39, no. 4, pp. 525–544, 2006.

[27] G. Anil and R. P. Wiegand, "Black-box search by elimination of fitness functions," in *Proc. 10th ACM SIGEVO Workshop Found. Genet. Algorithms (FOGA)*, Orlando, FL, USA, 2009, pp. 67–78.

[28] P. Afshani, M. Agrawal, B. Doerr, C. Doerr, K. G. Larsen, and K. Mehlhorn, "The query complexity of finding a hidden permutation," in *Discrete Applied Mathematics*. Heidelberg, Germany: Springer, 2019, pp. 28–50.

[29] B. Doerr, *Probabilistic Tools for the Analysis of Randomized Optimization Heuristics*. Cham, Switzerland: Springer, 2020, pp. 1–87. [Online]. Available: https://arxiv.org/abs/1801.06733

[30] G. R. Harik, F. G. Lobo, and D. E. Goldberg, "The compact genetic algorithm," *IEEE Trans. Evol. Comput.*, vol. 3, no. 4, pp. 287–297, Nov. 1999.

[31] P. K. Lehre and C. Witt, "Black-box search by unbiased variation," *Algorithmica*, vol. 64, no. 4, pp. 623–642, 2012.

[32] T. Jansen and C. Zarges, "Performance analysis of randomised search heuristics operating with a fixed budget," *Theor. Comput. Sci.*, vol. 545, pp. 39–58, Aug. 2014.

[33] B. Doerr, T. Jansen, C. Witt, and C. Zarges, "A method to derive fixed budget results from expected optimisation times," in *Proc. 15th Annu. Conf. Genet. Evol. Comput. (GECCO)*, 2013, pp. 1581–1588.

[34] B. Doerr and C. Winzen, "Ranking-based black-box complexity," *Algorithmica*, vol. 68, no. 3, pp. 571–609, 2014.

[35] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Berlin, Germany, 2017, pp. 777–784.

[36] D. Antipov and B. Doerr, "Precise runtime analysis for plateaus," in *Proc. Int. Conf. Parallel Problem Solving Nat. (PPSN)*, 2018, pp. 117–128.

[37] B. Doerr, C. Doerr, and J. Yang, "k-bit mutation with self-adjusting k outperforms standard bit mutation," in *Proc. Int. Conf. Parallel Problem Solving Nat. (PPSN)*, 2016, pp. 824–834.

[38] B. Doerr, A. Lissovoi, P. S. Oliveto, and J. A. Warwicker, "On the runtime analysis of selection hyper-heuristics with adaptive learning periods," in *Proc. Genet. Evol. Comput. Conf. (GECCO)*, Kyoto, Japan, 2018, pp. 1015–1022.

[39] B. Doerr and F. Neumann, *Theory of Evolutionary Computation—Recent Developments in Discrete Optimization*. Cham, Switzerland: Springer, 2020.

[40] P. S. Oliveto and C. Witt, "Simplified drift analysis for proving lower bounds in evolutionary computation," *Algorithmica*, vol. 59, no. 3, pp. 369–386, 2011.

[41] P. S. Oliveto and C. Witt, "Erratum: Simplified drift analysis for proving lower bounds in evolutionary computation," *CoRR*, vol. abs/1211.7184, 2012. [Online]. Available: http://arxiv.org/abs/1211.7184

[42] B. Doerr, "Analyzing randomized search heuristics via stochastic domination," *Theoretical Comput. Sci.*, vol. 773, pp. 115–137, Jun. 2019.

[43] C. Gießen and C. Witt, "The interplay of population size and mutation probability in the $(1+\lambda)$ EA on OneMax," *Algorithmica*, vol. 78, pp. 587–609, Jun. 2017.

[44] B. Doerr and D. Johannsen, "Edge-based representation beats vertex-based representation in shortest path problems," in *Proc. 12th Annu. Conf. Genet. Evol. Comput.*, 2010, pp. 759–766.

Benjamin Doerr received the Diploma, Ph.D., and Habilitation degrees in mathematics from the Christian-Albrechts-Universität zu Kiel, Kiel, Germany, in 1998, 2000, and 2005, respectively.

He is a Full Professor with the École Polytechnique, Palaiseau, France. From 2005 to 2013, he was a Tenured Senior Researcher with the Max Planck Institute for Informatics, Saarbräcken, Germany, and an Adjunct Professor with Saarland University, Saarbräcken. His research interests include mathematical analyzes of problem-specific algorithms and randomized search heuristics as well as the development of new algorithms based on such analyzes.

Prof. Doerr is an Editorial Board Member of several scientific journals, among them, *Evolutionary Computation*, *Natural Computing*, and *Artificial Intelligence Journal*.

Martin S. Krejca received the B.S. and M.S. degrees in computer science from the University of Jena, Jena, Germany, in 2012 and 2014, respectively, and the Ph.D. degree in computer science from the Hasso Plattner Institute, University of Potsdam, Potsdam, Germany, in 2019.

He is currently a Post-Doctoral Researcher with the Algorithm Engineering Group, Hasso Plattner Institute, University of Potsdam. His research interests include the theoretical analysis of randomized processes, especially the analysis of estimation-of-distribution algorithms, and the development of efficient tools for this goal.