The Competing Genes Evolutionary Algorithm: Avoiding Genetic Drift Through Competition, Local Search, and Majority Voting

Adetunji David Ajimakin and V. Susheela Devi

***Abstract*—An estimation-of-distribution algorithm is an evolutionary algorithm that uses a probabilistic model to represent its population. It iteratively draws solutions from its model and, considering their fitnesses, uses these solutions to update the model's parameters. However, there is a risk of reinforcing random noise from sampling back into the model—a problem termed genetic drift. We propose the competing genes evolutionary algorithm that optimizes a fitness function over a binary search space and avoids this problem by updating only one model parameter at a time. The algorithm uses the model not only to sample solutions but also to select a parameter in each iteration that it pins to a value for the rest of the search process. We obtain upper bounds on the number of fitness evaluations the algorithm needs to solve well-known benchmark functions as a function of its population size and observe better or comparable results to other evolutionary algorithms. We attribute these favorable results to the algorithm's efficient use of its samples to explore its dwindling search space. We also introduce two variants of the algorithm. The first version eliminates the need to preset the number of solutions to sample per iteration, and the second seeks to escape local optima. We provide evidence that these variants achieve their goals.**

***Index Terms*—Estimation-of-distribution algorithms (EDAs), Gauss–Southwell rule, genetic drift, local search, majority voting, natural gradient, runtime analysis.**

I. Introduction

An estimation-of-distribution algorithm (EDA) is an evolutionary algorithm that evolves a probabilistic model defined over the search space. The algorithm optimizes a fitness function by sampling solutions from its model and refining the model to improve the quality of solutions it produces in latter iterations. This process of sampling and refining repeats until it finds a solution of good-enough quality or exhausts its computational budget. The simplest EDAs use univariate models, which assume the problem variables are independent and factorize the joint distribution over the search space as a product of marginal distributions of the variables. Examples of univariate EDAs include the compact genetic algorithm (cGA) [1], the univariate marginal distribution algorithm (UMDA) [2], the population-based incremental learning (PBIL) [3], and the max–min ant system with iteration-best update (MMAS_{ib}) [4]. Although EDAs apply to general search spaces and may use complex models [5], our discourse is in the context of univariate EDAs that optimize pseudo-Boolean fitness functions defined on the set of binary strings of length n. Mathematical analyses of EDAs in this context have provided insights into the behavior of these algorithms, typically in terms of their runtime.

The runtime of an EDA is the number of fitness evaluations needed to sample an optimal solution for the first time. A univariate EDA's runtime on a problem depends on how quickly it can set the marginals to the correct values, maximizing the chances of producing good solutions. This runtime can become large if the univariate EDA experiences a phenomenon known as genetic drift, the tendency of the probability model to converge on a suboptimal solution without justification from the fitness function [6]. In detail, consider the UMDA, which represents its model as an n-dimensional probability vector. The components of this vector are called marginals or frequencies, and they encode the probability that a sampled solution has a 1 in the corresponding bit position. The UMDA samples λ solutions in each iteration and updates its probability vector to the relative frequency of 1s in the top μ solutions. Consider a bit position that is neutral in an iteration. This means it does not influence the fitness. The update to its marginal is, in essence, an empirical average of random samples drawn from a Bernoulli distribution. This empirical average is unlikely to be the same as the distribution's mean; hence the drift in the marginal until it eventually becomes either 0 or 1. Krejca [7] formalized this argument using the theory of Martingales under the framework of n-Bernoulli-λ-EDAs introduced in [8]. Another insight into this issue is Shapiro's work in [9], where he described an EDA as a Markov chain process on the space of its model parameters. He then argued that genetic drift results from the Markov chain's lack of ergodicity and irreducibility. An analysis of the UMDA in [10], coupled with the result in [11], shows that genetic drift can significantly slow down the algorithm when optimizing the DECEPTIVELEADINGBLOCKS benchmark function.

We can choose to avoid genetic drift or include mechanisms that afford corrections should drift occur. One such correction mechanism is the use of borders that prevent marginals from reaching the absorbing states of 0 and 1. The UMDA with

Manuscript received 1 May 2022; revised 3 October 2022, 22 November 2022, and 5 December 2022; accepted 9 December 2022. Date of publication 14 December 2022; date of current version 1 December 2023. *(Corresponding author: Adetunji David Ajimakin.)*

The authors are with the Department of Computer Science and Automation, Indian Institute of Science, Bengaluru 560012, India (e-mail: ajimakind@iisc.ac.in; susheela@iisc.ac.in).

Color versions of one or more figures in this article are available at https://doi.org/10.1109/TEVC.2022.3229038.

Digital Object Identifier 10.1109/TEVC.2022.3229038

borders optimizes the ONEMAX function with expected runtime $O(n \ln n)$ if it uses a population size of $\Theta(\ln n)$ [12], [13], even though it experiences a strong drift that marches its marginals to the wrong border in this regime [14]. Without borders, the UMDA requires a population size above $c\sqrt{n} \ln n$, for a large constant $c > 0$, to optimize ONEMAX. In this regime, the marginals will rarely go below $1/2$, but the algorithm still requires the use of borders to get a finite expected runtime of $O(n \ln n)$.

A reasonable idea central to some EDAs, such as PBIL, cGA, and MMAS$_{ib}$ is to make only slight changes to the model in each iteration. The PBIL replaces the update scheme of the UMDA with a convex combination of the probability vector and the empirical mean of the fittest solutions. The cGA samples two solutions, and in bit positions where the two solutions differ, changes the marginals by $\pm 1/K$ in the direction of the fitter solution. Here, K is called the hypothetical population size of the algorithm. The MMAS$_{ib}$ decreases all marginals (called pheromones) by a factor $(1-\rho)$ and adds ρ to the marginals of bit positions where the best solution has a 1. The learning rate of the PBIL, the evaporation factor ρ of the MMAS$_{ib}$, and the update strength $1/K$ of the cGA all play the same role in reducing genetic drift. Analyses of these EDAs [6], [15], [16] invariably lead to the same conclusion: these parameters must be large enough to make progress on the optimization problem but not so large that they reinforce random fluctuations due to sampling into the model.

The relationship between an EDA's update strength and its runtime on a problem can be complex. Analyses in [17] suggest the cGA's performance on the ONEMAX function is optimal in two regions of its parameter space and worse in the region between those two. Using the right update strength is essential, and while methods exist for parameter tuning and control [18], a common technique is to remove the dependency on the algorithm's parameters. For example, the parameterless hierarchical Bayesian optimization algorithm [19] and the parallel-run EDA [20] use multiple copies of the base algorithm running in parallel. Each copy has a different population size and a carefully chosen computational budget. Doerr [20] proved, under reasonable assumptions, that the parallel-run EDA has a logarithmic factor performance loss over the EDA using the optimal population size. Along the same lines, Doerr and Zheng [21] introduced the *smart-restart* cGA, whose hypothetical population size exponentially increases with each restart. The *smart-restart* cGA relies on the analyses in [6] to end a run by checking if the risk of genetic drift is too high. Doerr and Zheng [6] also listed for other univariate EDAs the time needed for the marginal of a neutral bit position to hit a boundary.

An alternative approach to avoid genetic drift is to bias the model toward the Uniform distribution. Shapiro [9] introduced a variant of the UMDA that satisfies, on a constant fitness function, the detailed balance condition from the theory of Markov Chains. Mühlenbein [22] proposed the use of priors in the update of EDAs. Friedrich et al. [8] proposed an EDA called scGA that introduces a bias into the update process to make the marginals concentrate around $1/2$ for neutral bit positions. However, Doerr and Krejca [23] showed that the scGA performs poorly on the ONEMAX function.

To overcome the effects of genetic drift, Doerr and Krejca [23] proposed a new EDA called the significance-based cGA (sig-cGA). The sig-cGA, like the cGA, samples two solutions in each iteration and saves the fitter solution into an archive. It changes the marginal of a bit position only if it detects a statistically significant abundance of either 1s or 0s in that position among the solutions stored in the archive. The algorithm optimizes ONEMAX and LEADINGONES both in time $O(n \ln n)$ with high probability, a feat that had not been proven for any other EA or EDA prior to that time.

Following the footsteps of the sig-cGA in leaving the marginals unchanged until there is a justification for making a change, we propose the competing genes evolutionary algorithm (cgEA). The algorithm uses the model not only to sample solutions but also to select, in each iteration, a marginal it pins to a value for the rest of the search process. Furthermore, it combines local search and a majority voting rule to determine the marginal of the selected bit position. This algorithm is similar to the significant bit voting algorithm [24] of Rowe and Aishwaryaprajna, but ours differs by its use of local search and its approach to finding the significant bit position. We prove that the cgEA optimizes any MONOTONE function in $O(n)$ evaluations, the LEADINGONES and DECEPTIVELEADINGBLOCKS functions in $O(n \ln n)$ evaluations, the JUMP$_k$ function in $O\left(4^k n \ln n\right)$ evaluations, and the JUMPOFFSETSPIKE$_k$ function in $O\left(e^k n^{1.5} \ln n\right)$ evaluations for optimal setting of the population size.

We show how a simple restart strategy [25], [26] with increasing population size removes the need for population-size tuning and results in only a constant factor performance loss over the cgEA using the optimal population size. We also introduce and empirically verify a version that escapes local optima by restarting with a biased initial distribution. We conclude this article by discussing the algorithm's limitations and further extensions.

II. PRELIMINARIES

We consider the maximization of pseudo-Boolean fitness functions $f: \{0,1\}^n \to \mathbb{R}$, where $\{0,1\}^n$ is the binary search space with dimension $n \in \mathbb{N}^+$. We call an element $x \in \{0,1\}^n$ an *individual*. The bit positions of x assume the role of genes, and we denote by x_i the bit in the ith position of x for an $i \in [n] = \{1, \ldots, n\}$. We denote the number of ones in x by $\|x\|_1$.

As mentioned earlier, the runtime of an algorithm optimizing a fitness function is the number of fitness evaluations the algorithm makes until it generates an optimal solution. In this work, we deal with upper bounds on the runtime that hold with high probability, i.e., the probability that the runtime is at most the stated value is at least $1 - o(1)$. Our preference for high-probability statements rather than statements about expectations stems from Doerr's arguments in [20], which showed how to turn a high-probability guarantee into a bound on the expected runtime.

A. Benchmark Functions

1) OneMax: The generalized $\textsc{OneMax}_z(x)$ function, for some fixed target string $z \in \{0,1\}^n$, tests the performance of an EA as a hill climber [25] by assigning to x the number of positions where x and z agree. Formally

$$\textsc{OneMax}_z(x) = |\{i \in [n] \mid x_i = z_i\}|.$$

OneMax is a member of the class of generalized Monotone functions.

Definition 1: A function $f : \{0,1\}^n \to \mathbb{R}$ is monotone with respect to $z \in \{0,1\}^n$ if for all $x, y \in \{0,1\}^n$ where $\{i \in [n] \mid x_i = z_i\} \subset \{i \in [n] \mid y_i = z_i\}$, then $f(x) < f(y)$.

2) LeadingOnes: The generalized LeadingOnes function tests the ability of an EA to cope with dependencies among variables [25] by applying a permutation σ to both x and a target string z and counting the number of positions where the rearrangements agree until the first disagreement. Formally

$$\textsc{LeadingOnes}_{z,\sigma}(x) = \max\left\{i \in \{0,\ldots,n\} \mid \forall j \in [i] : x_{\sigma(j)} = z_{\sigma(j)}\right\}.$$

For our analysis, we consider the instance of generalized LeadingOnes with the identity permutation and an all-ones target string. This instance counts the number of ones until the first zero in a bit string. The performance of the cgEA on this instance generalizes to the whole class because the cgEA is unbiased as it neither discriminates between the positions nor differentiates between the bits 1 and 0. The instance we analyze is

$$\textsc{LeadingOnes}(x) = \sum_{i=1}^{n} \prod_{j=1}^{i} x_j.$$

Doerr and Krejca [23] summarized the expected runtime of various EAs on the OneMax and LeadingOnes functions. They note that many of the algorithms have an expected runtime of $\Theta(n \ln n)$ on OneMax and $\mathrm{O}(n^2)$ on LeadingOnes for optimal parameter settings.

3) DeceptiveLeadingBlocks: Lehre and Nguyen [11] introduced the DeceptiveLeadingBlocks (DLB) problem as a deceptive version of LeadingOnes to study the behavior of the UMDA in the presence of deception and epistasis. The function takes in a bitstring $x \in \{0,1\}^n$, where n is an even positive integer, and pairs x_{2i-1} and x_{2i} for all $i \in [n/2]$ to form $n/2$ blocks. The active block is the leftmost block whose value is not 11. Suppose this is the $(k+1)$th block numbered from the left, then the fitness of x is n if there is no active block, $2k+1$ if the active block's value is 00, and $2k$ if 01 or 10. The function deceives an algorithm to prefer the locally optimal value of 00 for the active block even though the global optimum is the all-ones string. A formal definition of DLB is

$$\mathrm{DLB}(x) = \begin{cases} n, & \text{if } \phi(x) = \frac{n}{2} \\ 2\phi(x)+1, & \text{if } x_{2\phi(x)+1} + x_{2\phi(x)+2} = 0 \\ 2\phi(x), & \text{if } x_{2\phi(x)+1} + x_{2\phi(x)+2} = 1 \end{cases}$$

where

$$\phi(x) = \sum_{i=1}^{\frac{n}{2}} \prod_{j=1}^{2i} x_j$$

is the number of leading 11 s. Classical EAs optimize DLB in $\mathrm{O}(n^3)$ expected runtime [11]. Doerr and Krejca [10] proved that the UMDA can optimize DLB with high probability in $\mathrm{O}(n^2 \ln n)$ fitness evaluations if its parameters are in the regime that avoids genetic drift.

4) Jump: The $\textsc{Jump}_k$ function with gap size $k \in \mathbb{N}^+$ is a multimodal function to test the ability of an algorithm to escape local optima. It has a fitness landscape similar to that of OneMax with a target string of all-ones except for the introduction of a fitness valley of size $k \le n$ around the global optimum. Formally

$$\textsc{Jump}_k(x) = \begin{cases} k + \|x\|_1, & \text{if } \|x\|_1 \le n-k \text{ or } \|x\|_1 = n \\ n - \|x\|_1, & \text{otherwise.} \end{cases}$$

Doerr [20] proved that the cGA with high probability optimizes $\textsc{Jump}_k$ in $\mathrm{O}(n \ln n)$ if $k < ((\ln n)/20)$.

5) JumpOffsetSpike: Witt [27] noted that algorithms employing majority-vote crossover have an inherent bias to approach the all-ones string when optimizing the $\textsc{Jump}_k$ function. He modified the $\textsc{Jump}_k$ function by shifting the gap from the all-ones string and placing global optima midway within the gap. This new problem is called the $\textsc{JumpOffsetSpike}_k$ function or JOS_k for short. For the problem to be well defined, the problem dimension n must be a multiple of 4, and the gap size $k \le n/4$ must be an even positive integer. A formal definition of the function is

$$\mathrm{JOS}_k(x) = \begin{cases} k + \|x\|_1, & \text{if } \|x\|_1 \le \frac{3n}{4} \\ & \quad \text{or } \|x\|_1 \ge \frac{3n}{4} + k \\ n + k + 1, & \text{if } \|x\|_1 = \frac{3n}{4} + \frac{k}{2} \\ \frac{3n}{4} + k - \|x\|_1, & \text{otherwise.} \end{cases}$$

Analyses in [27] showed that majority-vote crossover is ineffective on this problem. However, the cGA using a hypothetical population size of $\lceil c\sqrt{n} \ln n \rceil$ for a sufficiently large constant $c > 0$ finds an optimal solution with a high probability in runtime $O(n \ln n)$ if the gap size $k = O(\sqrt{n}/\ln^6 n)$.

6) Maximum Independent Vertex Set: An independent vertex set of a graph $G = (V, E)$ is a subset S of V such that no two vertices in S form an edge in E. The maximum independent vertex set problem (MIVS) [28, function F22] is to find an independent vertex set of the largest size. Under the interpretation that $i \in S$ if and only if $x_i = 1$, a formal definition of MIVS is

$$\mathrm{MIVS}(x) = \sum_{i=1}^{n} x_i - n \sum_{(i,j) \in E} x_i x_j.$$

Here, we consider the undirected graph $G = (V, E)$ where $V = \{1,\ldots,n\}$ and

$$\begin{aligned} (i,j) \in E \text{ if } & j = i+1 \text{ when } i \ne \frac{n}{2} \\ \text{or } & j = i+1+\frac{n}{2} \text{ when } i < \frac{n}{2} \\ \text{or } & j = i-1+\frac{n}{2} \text{ when } 2 \le i \le \frac{n}{2}. \end{aligned}$$

The maximum value of MIVS on this graph is $n/2$ if n is a multiple of 4; otherwise, it is $n/2+1$. We require that $n \ge 4$ and is even.

7) Ising Models: We consider two problems with origins in statistical mechanics. Let E be a set of pairs such that if $(i, j) \in E$, then x_i and x_j are neighbors. We seek to maximize the following objective function:

$$\textsc{Ising}(x \mid E) = \sum_{(i,j)\in E} x_i x_j + (1 - x_i)\big(1 - x_j\big)$$

which attains its maximum when neighboring variables have the same value. The two instances we focus on differ by their neighborhood structure. They are as follows.

1) IsingRing [28, function F19] which arranges the variables to form a 1-*D* ring. Variables x_i and x_j are neighbors if

$$j = (i \bmod n) + 1 \text{ or } i = (j \bmod n) + 1.$$

This instance has a maximum value of n.

2) IsingTorus [28, function F20] which arranges the variables into a 2-*D* square lattice that wraps around in both dimensions. This arrangement results in each variable having four neighbors. Formally, x_i and x_j are neighbors if

$$c(j) = (c(i) \pm 1) \bmod \sqrt{n} \quad \text{when } r(i) = r(j)$$

or

$$r(j) = (r(i) \pm 1) \bmod \sqrt{n} \quad \text{when } c(i) = c(j).$$

This instance has a maximum value of $2n$ with n being a perfect square. The helper functions $r(k)$ and $c(k)$ extract, respectively, the row and column indices into the lattice for bit position k. They are defined as $r(k) = \lfloor((k-1)/\sqrt{n})\rfloor$ and $c(k) = (k-1) \bmod \sqrt{n}$.

B. Tools for Analysis

Our analyses of the cgEA on the benchmark functions routinely use concentration inequalities and other elementary results that we state here for easy reference.

The first is the following well-known additive Chernoff bound due to Hoeffding [29] that gives an exponential bound on the probability that the difference between the sum of bounded independent random variables and its expected value exceeds a certain amount. We state the inequalities as given in [30, Th. 1.10.9].

Lemma 1: Let $X_1, \ldots, X_n$ be independent random variables. Assume that each X_i takes values in a real interval $[a_i, b_i]$ of length $c_i = b_i - a_i$. Let $X = \sum_{i=1}^n X_i$. Then, for all $t > 0$

$$\Pr[X \geq \mathrm{E}[X] + t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right)$$

and

$$\Pr[X \leq \mathrm{E}[X] - t] \leq \exp\left(-\frac{2t^2}{\sum_{i=1}^n c_i^2}\right).$$

The next lemma from [30, Th. 1.10.21] is a multiplicative Chernoff bound for deviations above the expectation.

Lemma 2: Let $X_1, \ldots, X_n$ be independent random variables taking values in $[0, 1]$. Let $X = \sum_{i=1}^n X_i$. Let $\mathrm{E}[X] \leq \mu^+$. Then, for all $\delta \in [0, 1]$

$$\Pr\big[X \geq (1+\delta)\mu^+\big] \leq \exp\left(-\frac{\delta^2\mu^+}{3}\right).$$

We use the following lemma from [30, Lemma 1.7.2] to correct for the probability of an event when we can condition its occurrence on another event that happens almost surely.

Lemma 3: Let C be some event with probability $1 - p$. Let A be any event. Then

$$\frac{\Pr[A] - p}{1 - p} \leq \Pr[A \mid C] \leq \frac{\Pr[A]}{1 - p}.$$

If $p \leq (1/2)$, then $\Pr[A] - p \leq \Pr[A \mid C] \leq \Pr[A] + 2p$.

Finally, we prove a valuable estimate for deviations from the central binomial coefficient.

Lemma 4: Let $m \in \mathbb{N}^+$ and $d \in \{0, \ldots, m\}$. Then

$$\left(\frac{m-d}{m}\right)^d \frac{4^m}{\sqrt{4m}} \leq \binom{2m}{m+d} < \left(\frac{m}{m+d}\right)^d \frac{4^m}{\sqrt{\pi m}}.$$

Proof: Sasvári [31] gave the following bounds on the central binomial coefficient:

$$\binom{2m}{m} > \frac{4^m}{\sqrt{\pi m}} \exp\left(-\frac{1}{8m}\right)$$

and

$$\binom{2m}{m} < \frac{4^m}{\sqrt{\pi m}} \exp\left(-\frac{1}{8m} + \frac{1}{192m^3}\right).$$

From these, it follows that:

$$\binom{2m}{m} < \frac{4^m}{\sqrt{\pi m}} \exp\left(-\frac{1}{8m} + \frac{1}{192m^3}\right) < \frac{4^m}{\sqrt{\pi m}}$$

and

$$\begin{aligned}\binom{2m}{m} &> \frac{4^m}{\sqrt{\pi m}} \exp\left(-\frac{1}{8m}\right) = \frac{4^m}{\sqrt{4\ \mathrm{m}}} \frac{2}{\sqrt{\pi}} \exp\left(-\frac{1}{8m}\right) \\ &> \frac{4^m}{\sqrt{4m}} \quad \text{for} \quad m \geq 2 > \frac{1}{8\ln\left(2/\sqrt{\pi}\right)}.\end{aligned}$$

$$\binom{2m}{m} = \frac{4^m}{\sqrt{4m}} \text{ by inspection when } m = 1.$$

Together, we obtain the following simplified bounds:

$$\frac{4^m}{\sqrt{4m}} \leq \binom{2m}{m} < \frac{4^m}{\sqrt{\pi m}}. \tag{1}$$

Define a ratio R as

$$R = \frac{\binom{2m}{m+d}}{\binom{2m}{m}} = \frac{m!}{(m-d)!} \cdot \frac{m!}{(m+d)!} = \prod_{i=1}^{d} \frac{m-d+i}{m+i}.$$

We have

$$\left(\frac{m-d}{m}\right)^d \leq R \leq \left(\frac{m}{m+d}\right)^d. \tag{2}$$

We combine Bounds (1) and (2) to conclude the proof. ■

III. Competing Genes Evolutionary Algorithm

The cgEA is a univariate EDA that represents its probability model as an n-dimensional vector p. Each component or marginal $p_i \in [0, 1]$ for $i \in [n]$ is the probability of sampling a 1 bit at position i independently of other positions. Thus, it samples any individual $x = (x_1, \ldots, x_n)$ with joint probability

$$\pi(x \mid p) = \prod_{i=1}^{n} {p_i}^{x_i}(1 - p_i)^{(1-x_i)}.$$

The cgEA proceeds in three phases per iteration for n iterations to maximize a pseudo-Boolean fitness function f.

In the first phase, the cgEA samples a set X of λ individuals according to $\pi(\cdot \mid p)$ and assumes that there is a bit position with the most significant influence on the fitness of the individuals in X. We assume this position has the largest (in absolute value) rate of change in the expected fitness under the current distribution. While this criterion suggests we use the gradient, we instead opt for the natural gradient [32], [33] as it considers the Riemannian structure of the parameter space to adjust the standard gradient [34]. Moreover, the use of the natural gradient is central to Natural Evolution Strategies [35] and Information Geometric Optimization [36]. Let

$$J(p) = \mathrm{E}\big[f(x)\big] = \sum_{x \in \{0,1\}^n} f(x)\pi(x \mid p)$$

be the expected fitness under the current distribution.

For all $i \in [n]$

$$\begin{aligned}
\nabla_{p_i} J(p) &= \nabla_{p_i} \sum_{x \in \{0,1\}^n} f(x)\pi(x \mid p) \\
&= \sum_{x \in \{0,1\}^n} f(x)\nabla_{p_i}\pi(x \mid p) \\
&= \sum_{x \in \{0,1\}^n} f(x)\nabla_{p_i}\pi(x \mid p)\frac{\pi(x \mid p)}{\pi(x \mid p)} \\
&= \sum_{x \in \{0,1\}^n} \big[f(x)\nabla_{p_i} \ln \pi(x \mid p)\big]\pi(x \mid p)
\end{aligned}$$

by an application of the log-derivative "trick". So

$$\begin{aligned}
\nabla_{p_i} J(p) &= \mathrm{E}\big[f(x)\nabla_{p_i} \ln \pi(x \mid p)\big] \\
&= \mathrm{E}\left[f(x)\left(\frac{x_i}{p_i} - \frac{1 - x_i}{1 - p_i}\right)\right].
\end{aligned}$$

Let $\widetilde{\nabla}_{p_i} J(p)$ be the natural gradient of $J(p)$ with respect to p_i. By definition in [32]

$$\widetilde{\nabla}_{p_i} J(p) \stackrel{\text{def}}{=} F_{p_i}^{-1} \nabla_{p_i} J(p)$$

where $F_{p_i} = [1/(p_i(1 - p_i))]$ is the Fisher Information of a Bernoulli random variable with mean p_i. We obtain

$$\widetilde{\nabla}_{p_i} J(p) = \mathrm{E}\big[f(x)\,((1 - p_i)x_i - p_i(1 - x_i))\big].$$

We seek the position i that maximizes $|\widetilde{\nabla}_{p_i} J(p)|$ and resort to a Monte Carlo estimate of the expectation using the sampled individuals

$$\big|\widetilde{\nabla}_{p_i} J(p)\big| \approx \left|\frac{1}{\lambda}\sum_{x \in X} f(x)[(1 - p_i)x_i - p_i(1 - x_i)]\right|.$$

As it stands, the position selection is not invariant to monotonic transformations of the fitness function. To remedy that, we replace the fitness value of $x \in X$ with a utility $U_X(x)$ that depends only on the fitness rank of x. Our choice of the utility function is

$$U_X(x) = \frac{1}{|\{x' \in X \mid f(x') \geq f(x)\}|}$$

i.e., the utility of x is inversely proportional to the number of solutions fitter than or equally as fit as x. We call the resulting simplified estimate the Gauss–Southwell score of position i, $\mathrm{GS}(i)$, as the selection heuristic is reminiscent of the Gauss–Southwell rule [37] in coordinate descent methods. Explicitly

$$\mathrm{GS}(i) = \left|(1 - p_i)\sum_{\substack{x \in X \\ x_i = 1}} U_X(x) - p_i \sum_{\substack{x \in X \\ x_i = 0}} U_X(x)\right|.$$

We view the first phase as a competition between the positions. The winning position is the one with the largest Gauss–Southwell score. This position determines the scope of the local search in the second phase, and the algorithm updates the position's marginal by a majority vote in the third phase.

In the second phase (lines 6–10 of Algorithm 1), the cgEA creates a second population Y from the first population X through local search. Specifically, it creates for each $x \in X$ a twin x' that differs only in the winning position. It then moves the fitter of the two into the second population. If the selected marginal has low variance, then without the local search phase, there will not be enough diversity in that position to effect a change via a majority vote.

In the third and final phase, the algorithm sets the marginal of the winning position to the most occurring value (breaking ties uniformly at random) in that position among the $\lfloor \lambda/2 \rfloor$ best individuals in the second population. See Algorithm 1 for a pseudocode description.

A top-down view of the algorithm is that it splits the search space in each iteration and chooses the promising region containing the best individuals. As a result, it efficiently uses its samples to explore its shrinking search space. An implementation may switch to exhaustive enumeration when the number of unset marginals is at most $\log_2 \lambda$ to reduce the runtime.

IV. Runtime Analyses

In the following analyses, we are interested in the probability that the algorithm samples the optimum in at most n iterations as a function of the population size λ. We assume that the Uniform distribution is the starting distribution.

A. Analysis on Monotone Functions

Theorem 1: Let $f\colon \{0, 1\}^n \to \mathbb{R}$ be an n-dimensional monotone function with respect to $z \in \{0, 1\}^n$. The cgEA with population size $\lambda \geq 2$ and a uniform starting distribution optimizes f in at most $2n\lambda$ function evaluations.

Proof: Suppose the cgEA selects the ith position in the first phase in an iteration. It follows from Definition 1 of Monotone functions that the twin whose bit in the ith position is equal to z_i goes into the second population Y.

Algorithm 1: cgEA With Population Size $\lambda \in \mathbb{N}$, $\lambda \geq 2$ and Starting Distribution p to Maximize $f\colon \{0,1\}^n \to \mathbb{R}$

$\mu \leftarrow \lfloor \lambda/2 \rfloor$
Let $z_{best} \in \{0,1\}^n$ be drawn according to $\pi(\cdot \mid p)$.
repeat
  if *all marginals are either* 0 *or* 1 **then**
    return z_{best}
  Let X be a set of λ individuals sampled according to $\pi(\cdot \mid p)$.
  $i^* \leftarrow \arg\max_{i \in \{1,\ldots,n\}} GS(i)$
  $Y \leftarrow \emptyset$
  foreach $x \in X$ **do**
    Let x' be a copy of x with bit i^* flipped.
    Let y be the fitter individual between x and x' (breaking ties uniformly at random).
    $Y \leftarrow Y \cup \{y\}$
  Let $z^{(1)}, \ldots, z^{(\mu)}$ denote the μ best individuals in Y in descending order of fitness (breaking ties uniformly at random).
  if $f(z^{(1)}) \geq f(z_{best})$ **then** $z_{best} \leftarrow z^{(1)}$
  $V \leftarrow \sum_{j=1}^{\mu} z_{i^*}^{(j)}$
  if $V > \mu/2$ **then** $p_{i^*} \leftarrow 1$
  else if $V < \mu/2$ **then** $p_{i^*} \leftarrow 0$
  else $p_{i^*} \leftarrow 1$ or 0 with equal probability
until *termination criterion is met*
return z_{best}

Consequently, the marginal of the ith position is set to z_i. In the nth iteration, all individuals resulting from the local search phase are equal to z. ■

B. Analysis on LeadingOnes

Theorem 2: Let $\varepsilon > 0$ be a constant. The cgEA with population size $\lambda > 68(2 + (\varepsilon + 2)\ln n)$ and a uniform starting distribution samples the optimum of the n-dimensional LeadingOnes function with probability at least $1 - n^{-\varepsilon}$ in at most $2n\lambda$ function evaluations.

Proof: The main idea is to show that the marginals are set to 1 from left to right with high probability. We call the leftmost bit position whose marginal is not 1 the active position. For convenience, assume the ith position is the active position. If the cgEA selects the active position, the local search ensures the ith bit of all individuals in the second population is 1. Consequently, the algorithm sets the marginal of the active position to 1 by a majority vote. We now only need to show that the algorithm, with high probability, selects the active position in each iteration.

Let X be the set of λ sampled individuals. Denote by R the random variable counting the number of individuals in X with a 1 in the active position, and by r its realization in an iteration. Let $\widetilde{X} = (\widetilde{X}_1, \ldots, \widetilde{X}_\lambda)$ be the sequence of elements in X in descending order of fitness, and I be the set of positions where the marginal is $1/2$.

Assume $r > 0$. For any individual $x \in X$ with 0 in the active position, its utility $U_X(x) = 1/\lambda$. All individuals with a 1 in the active position will not necessarily have the same fitness, so for an individual $x \in X$ with 1 in the active position, its utility $U_X(x) \geq 1/r$. Let u_{acc} be the accumulation of excess utility values beyond $1/r$ for individuals with 1 in the active position, i.e.,

$$u_{\text{acc}} = \sum_{j=1}^{r} \left(U_X(\widetilde{X}_j) - \frac{1}{r} \right).$$

Then, the Gauss–Southwell score of the active position is

$$\text{GS}(i) = \frac{1}{2} \left| u_{\text{acc}} + \frac{r}{r} - \frac{\lambda - r}{\lambda} \right| = \frac{u_{\text{acc}}}{2} + \frac{r}{2\lambda}.$$

Next, we derive an upper bound for the Gauss–Southwell score of any other position $k \in I \setminus \{i\}$. Let $Z_{jk} = 2\widetilde{X}_{jk} - 1$ for all $(j,k) \in [\lambda] \times [n]$, then

$$\begin{aligned}
\text{GS}(k) &= \frac{1}{2} \left| \sum_{j=1}^{\lambda} U_X(\widetilde{X}_j) Z_{jk} \right| \\
&= \frac{1}{2} \left| \sum_{j=1}^{r} \left(U_X(\widetilde{X}_j) - \frac{1}{r} + \frac{1}{r} \right) Z_{jk} + \frac{1}{\lambda} \sum_{j=r+1}^{\lambda} Z_{jk} \right| \\
&\leq \frac{u_{\text{acc}}}{2} + \frac{1}{2} \left| \frac{1}{r} \sum_{j=1}^{r} Z_{jk} + \frac{1}{\lambda} \sum_{j=r+1}^{\lambda} Z_{jk} \right|.
\end{aligned}$$

For the algorithm to select the active position, it is necessary that $\text{GS}(k) < \text{GS}(i)$ or sufficient that the inequality

$$\left| \frac{1}{r} \sum_{j=1}^{r} Z_{jk} + \frac{1}{\lambda} \sum_{j=r+1}^{\lambda} Z_{jk} \right| < \frac{r}{\lambda} \tag{3}$$

holds for all $k \in I \setminus \{i\}$.

Intuitively, we expect the right-hand side of inequality (3) to be concentrated around $1/2$ and the left-hand side around 0. We formalize this argument using Chernoff's bound.

Let S_k denote the argument of the absolute value function on the left-hand side of inequality (3). Then, $\text{E}[S_k] = 0$ because each Z_{jk} takes value in the set $\{-1, +1\}$ with equal probability. Also note that each summand in S_k is in the interval $[-1/r, 1/r]$. We apply Lemma 1 to obtain

$$\Pr\left[|S_k| < \frac{r}{\lambda} \right] > 1 - 2\exp\left(-\frac{r^4}{2\lambda^3} \right).$$

Given that R follows a binomial distribution with parameters λ and $1/2$, by an application of Lemma 1, we have

$$\Pr\left[R > \frac{(1-\delta)}{2}\lambda \right] > 1 - \exp\left(-\frac{\delta^2}{2}\lambda \right)$$

for $\delta \in [0,1]$. Therefore

$$\Pr\left[|S_k| < \frac{r}{\lambda} \;\middle|\; R > \frac{1-\delta}{2}\lambda \right] > 1 - 2\exp\left(-\frac{(1-\delta)^4}{32}\lambda \right).$$

By Lemma 3, we have

$$\Pr\left[|S_k| < \frac{r}{\lambda}\right] > 1 - 2\exp\left(-\tfrac{(1-\delta)^4}{32}\lambda\right) - 2\exp\left(-\tfrac{\delta^2}{2}\lambda\right) > 1 - 4\exp\left(-\tfrac{\lambda}{68}\right)$$

for $\delta = 0.1715$.

The probability that the cgEA selects the active position in each iteration is at least

$$1 - 4n^2\exp\left(-\frac{\lambda}{68}\right)$$

via Bernoullis's inequality for n positions for n iterations. Hence, for $\lambda > 68(2 + (\varepsilon + 2)\ln n)$, the algorithm solves LEADINGONES with probability $1 - n^{-\varepsilon}$ in at most $2n\lambda$ function evaluations. ■

C. Analysis on DECEPTIVELEADINGBLOCKS

Theorem 3: Let $\varepsilon > 0$ be a constant. The cgEA with population size $\lambda > 288(3 + (\varepsilon + 2)\ln n)$ and a uniform starting distribution samples the optimum of the n-dimensional DLB function with probability at least $1 - n^{-\varepsilon}$ in at most $2n\lambda$ function evaluations.

Proof: Our analysis of the cgEA on DLB is similar to that on LEADINGONES, to wit: we show the algorithm solves the blocks from left to right with high probability. By solving a block, we mean setting the marginals forming the block to 1 in any order. We call the first block whose marginals are not 1 the active block. Let the positions i and $i+1$ be the positions in the active block.

As is the case with LEADINGONES, let X be the set of λ sampled individuals, $\widetilde{X} = (\widetilde{X}_1, \ldots, \widetilde{X}_\lambda)$ be the sequence of elements in X in descending order of fitness, and I be the set of positions where the marginal is $1/2$. Let R_{ab} for all $(a, b) \in \{0, 1\}^2$ be the random variable denoting the number of individuals in X where the active block has value ab, and let r_{ab} be its realization in an iteration, i.e.,

$$r_{ab} = |\{x \in X \mid x_i = a, x_{i+1} = b\}|.$$

For any individual $x \in X$ with value 11 in the active block, its utility $U_X(x) \geq 1/r_{11}$. If the value were 00, then $U_X(x) = 1/(r_{11} + r_{00})$. For values 01 and 10, $U_X(x) = 1/\lambda$. Let u_{acc} be the accumulation of excess utility values beyond $1/r_{11}$ for individuals with 11 in the active block, i.e.,

$$u_{\text{acc}} = \sum_{j=1}^{r_{11}}\left(U_X(\widetilde{X}_j) - \frac{1}{r_{11}}\right).$$

Then, the Gauss–Southwell score of positions in the active block are

$$\text{GS}(i) = \frac{1}{2}\left|u_{\text{acc}} + \frac{r_{11}}{r_{11}} + \frac{r_{10}}{\lambda} - \frac{r_{01}}{\lambda} - \frac{r_{00}}{r_{11} + r_{00}}\right|$$

and

$$\text{GS}(i+1) = \frac{1}{2}\left|u_{\text{acc}} + \frac{r_{11}}{r_{11}} + \frac{r_{01}}{\lambda} - \frac{r_{10}}{\lambda} - \frac{r_{00}}{r_{11} + r_{00}}\right|.$$

Let us assume, without loss of generality, that $r_{10} \geq r_{01}$ to give $\text{GS}(i) \geq \text{GS}(i+1)$. If the algorithm selects position i in the competition phase, r_{11} individuals with 11 in the active block and the twins of r_{01} individuals with 01 in the active block will go into the second population Y in the local search phase. If, in addition, $r_{01} + r_{11} > \lambda/4$, then the algorithm sets the marginal of position i to 1.

We need to show that, with high probability, $r_{01} + r_{11} > \lambda/4$ and $\text{GS}(k) < \text{GS}(i)$ for all $k \in I \setminus \{i, i+1\}$. Under the assumption that $r_{10} \geq r_{01}$, we have

$$\text{GS}(i) \geq \frac{u_{\text{acc}}}{2} + \frac{r_{11}}{2(r_{11} + r_{00})}.$$

An upper bound of the Gauss–Southwell score of any other position $k \in I \setminus \{i, i+1\}$ is

$$\text{GS}(k) \leq \frac{u_{\text{acc}} + |S_k|}{2}$$

where

$$S_k = \frac{1}{r_{11}}\sum_{j=1}^{r_{11}} Z_{jk} + \frac{1}{r_{11} + r_{00}}\sum_{j=r_{11}+1}^{r_{11}+r_{00}} Z_{jk} + \frac{1}{\lambda}\sum_{j=r_{11}+r_{00}+1}^{\lambda} Z_{jk}$$

and

$$Z_{jk} = 2\widetilde{X}_{jk} - 1 \text{ for all } (j, k) \in [\lambda] \times [n].$$

Note that $\text{E}[S_k] = 0$ because each Z_{jk} takes value in the set $\{-1, +1\}$ with equal probability. Thus

$$\Pr[\text{GS}(k) < \text{GS}(i)] \geq \Pr\left[|S_k| < \frac{r_{11}}{r_{11} + r_{00}}\right] > 1 - 2\exp\left(-\frac{r_{11}^4}{2\lambda(r_{11} + r_{00})^2}\right)$$

via Lemma 1.

The random variable $R_{11} + R_{00}$ follows the binomial distribution $\text{Bin}(\lambda, 1/2)$, whereas R_{11} and R_{01} have conditional distributions of $\text{Bin}(r_{11} + r_{00}, 1/2)$ and $\text{Bin}(\lambda - r_{11} - r_{00}, 1/2)$, respectively. Consider an event $C = C_1 \cap C_2 \cap C_3$ for $\delta \in [0, 1]$ where

$$C_1 : R_{11} + R_{00} \in \left[(1-\delta)\frac{\lambda}{2},\ (1+\delta)\frac{\lambda}{2}\right]$$
$$C_2 : R_{11} > (1-\delta)\frac{r_{11} + r_{00}}{2}, \text{ and}$$
$$C_3 : R_{01} > (1-\delta)\frac{\lambda - r_{11} - r_{00}}{2}.$$

Let $a = \lceil((1-\delta)\lambda/2)\rceil$, and $b = \lfloor((1+\delta)\lambda/2)\rfloor$. Consider the partition of C_1 into disjoint events D_j for $j \in [a, b] \cap \mathbb{N}$, where D_j is the event that $R_{11} + R_{00} = j$. The events C_1, C_2, and C_3 are conditionally independent given D_j. By the law of total probability, we have

$$\Pr[C] = \sum_{j=a}^{b}\Pr\left[C_1 \mid D_j\right] \cdot \Pr\left[C_2 \mid D_j\right] \cdot \Pr\left[C_3 \mid D_j\right] \cdot \Pr\left[D_j\right].$$

Note that for all $j \in [a, b] \cap \mathbb{N}$

$$\begin{aligned}
\Pr[C_1 \mid D_j] &= 1\\
\Pr[C_2 \mid D_j] &\geq 1 - \exp\left(-\frac{\delta^2}{2}j\right) \text{ via Lemma 1}\\
&\geq \Pr[C_2 \mid D_a]\\
&\geq 1 - \exp\left(-\frac{\delta^2(1-\delta)}{4}\lambda\right) \geq 0
\end{aligned}$$

by substituting $((1-\delta)\lambda/2)$ for j.

Let E_1 be the event $R_{01} \leq ((1-\delta)(\lambda - j)/2)$. We assumed $r_{10} \geq r_{01}$. This is equivalent to assuming $r_{01} \leq ((\lambda - r_{11} - r_{00})/2)$, so let E_2 be the event $R_{01} \leq ((\lambda - j)/2)$, whose probability is at least $1/2$. Then

$$\begin{aligned}
\Pr[C_3 \mid D_j] &\geq 1 - \Pr[E_1 \mid E_2]\\
&= 1 - \frac{\Pr[E_2 \mid E_1]}{\Pr[E_2]} \cdot \Pr[E_1]\\
&> 1 - 2\Pr[E_1]\\
&\geq 1 - 2\exp\left(-\frac{\delta^2}{2}(\lambda - j)\right) \text{ via Lemma 1}\\
&\geq \Pr[C_3 \mid D_b]\\
&\geq 1 - 2\exp\left(-\frac{\delta^2(1-\delta)}{4}\lambda\right)
\end{aligned}$$

by substituting $((1+\delta)\lambda/2)$ for j. Lastly

$$\begin{aligned}
\sum_{j=a}^{b} \Pr[D_j] = \Pr[C_1] &\geq 1 - 2\exp\left(-\frac{\delta^2}{2}\lambda\right) \text{ via Lemma 1}\\
&\geq 1 - 2\exp\left(-\frac{\delta^2(1-\delta)}{4}\lambda\right).
\end{aligned}$$

By putting them together, we obtain

$$\begin{aligned}
\Pr[C] &> \Pr[C_2 \mid D_a] \cdot \Pr[C_3 \mid D_b] \cdot \sum_{j=a}^{b} \Pr[D_j]\\
&\geq 1 - 5\exp\left(-\frac{\delta^2(1-\delta)}{4}\lambda\right)
\end{aligned}$$

through a double application of the inequality $(1-x)(1-y) \geq 1-(x+y)$ for non-negative x and y. Let us condition on event C. We have

$$\frac{r_{11}^4}{2\lambda(r_{11}+r_{00})^2} > \frac{(1-\delta)^6}{128}\lambda$$

and

$$r_{01} + r_{11} > \frac{(1-\delta)}{2}\lambda.$$

Therefore

$$\Pr[\mathrm{GS}(k) < \mathrm{GS}(i) \mid C] > 1 - 2\exp\left(-\frac{(1-\delta)^6}{128}\lambda\right).$$

By applying Lemma 3, we obtain

$$\begin{aligned}
\Pr[\mathrm{GS}(k) < \mathrm{GS}(i)] &> 1 - 2\exp\left(-\frac{(1-\delta)^6}{128}\lambda\right)\\
&\quad - 10\exp\left(-\frac{\delta^2(1-\delta)}{4}\lambda\right)\\
&> 1 - 12\exp\left(-\frac{\lambda}{288}\right)
\end{aligned}$$

for $\delta = 0.126$. The probability that the cgEA selects a position in the active block is at least

$$1 - 12n\exp\left(-\frac{\lambda}{288}\right) \text{ via Bernoulli's inequality.}$$

If it does, the algorithm then sets the marginal of position i to 1 since $r_{11} + r_{01} > (\lambda/4)$ when $\delta = 0.126$.

The next iteration is, in essence, an iteration of the LEADINGONES problem, which we know the cgEA solves with probability at least $1 - 4n\exp(-(\lambda/68))$. Let F be the event that the cgEA solves a block of the DLB function. Then

$$\begin{aligned}
\Pr[F] &\geq \left(1 - 12n\exp\left(-\frac{\lambda}{288}\right)\right)\left(1 - 4n\exp\left(-\frac{\lambda}{68}\right)\right)\\
&> 1 - 16n\exp\left(-\frac{\lambda}{288}\right).
\end{aligned}$$

Therefore, the probability that the cgEA solves all $n/2$ blocks is at least

$$1 - 8n^2\exp\left(-\frac{\lambda}{288}\right)$$

via Bernoulli's inequality.

Hence, for $\lambda > 288(3 + (\varepsilon + 2)\ln n)$, the algorithm solves DLB with probability $1 - n^{-\varepsilon}$ in at most $2n\lambda$ function evaluations. ■

D. Analysis on JUMP

Theorem 4: Let $c > 0$ be a sufficiently large constant and $k \leq n/2 - 2$. The cgEA with population size $\lambda > c\,4^k \ln n$ and a uniform starting distribution optimizes the n-dimensional JUMP_k function with probability $1 - n^{-\Omega(1)}$ in at most $2n\lambda$ function evaluations.

Proof: We show that for the first $n - 2k - 4$ iterations, the cgEA with high probability sets no marginal to 0 and samples the optimum in the next iteration. Note that for the JUMP and JUMPOFFSETSPIKE functions, the competition phase only serves to select a position. Knowing that the selected position maximizes the Gauss–Southwell score is irrelevant.

In the local search phase, a bit flip changes the Hamming weight of an individual by ± 1. Therefore, we pessimistically assume that individuals in the first population with Hamming weights in the interval $[n-k,\ n-1]$ will, directly or indirectly through their twin, vote to assign 0 to the marginal of the selected position.

Let t denote the number of marginals already set by the algorithm and suppose the set X of sampled individuals in the current iteration $t+1$ does not contain an optimal individual. We partition X as follows:

$$\begin{aligned}
A &= \{x \in X \mid \|x\|_1 = n - k\}\\
B &= \{x \in X \mid 0 \leq \|x\|_1 < n - k\}\\
C &= \{x \in X \mid n - k < \|x\|_1 < n\}.
\end{aligned}$$

Let L_A, L_B, and L_C be random variables denoting the cardinality of the sets A, B, and C, respectively. Observe that the members of the set A are fitter than those in either B or C, and members of the set B are fitter than those in C. To ensure no marginal is set to 0, we require $L_A < \lambda/4$ and $L_B > \lambda/4$ so

that type B individuals form the majority even when all type A individuals are in the second population Y. These requirements imply $L_C < (3\lambda/4) - L_A$.

Let $q_A(t)$ be the probability that the algorithm samples an individual x with $\|x\|_1 = n-k$ in the current iteration $t+1$. We have

$$q_A(t) = \binom{n-t}{n-t-k} 2^{-(n-t)}.$$

Note that

$$q_A(t+1) = 2\binom{(n-t)-1}{(n-t-k)-1} 2^{-(n-t)}$$

and from the identity $\binom{v}{w} = \binom{v-1}{w} + \binom{v-1}{w-1}$, the inequality $\binom{v}{w} \leq 2\binom{v-1}{w-1}$ holds if $w \geq v/2$. Substituting $n-t$ for v and $n-t-k$ for w, we see that $q_A(t)$ is an increasing function of t when $t \leq n-2k-1$. Let $r = k+2$. Lemma 4 dictates that at $t = n-2k-4$, we have

$$q_A(n-2k-4) = \binom{2r}{r+2} 4^{-r} \leq \left(\frac{r}{r+2}\right)^2 \frac{1}{\sqrt{r\pi}} < 0.13$$

for all $r > 0$.

Given that L_A follows a binomial distribution with mean at most 0.13λ, we apply Lemma 2 to show that

$$\Pr\left[L_A < \frac{\lambda}{4}\right] \geq 1 - \exp\left(-\frac{12\lambda}{325}\right)$$

for $t \leq n-2k-4$.

Let us condition on the event $L_A < \lambda/4$, and let l_A be the realization of L_A. Denote by $q_C(t)$ the probability that the algorithm samples an individual x with $\|x\|_1 \in [n-k+1, n-1]$ in the current iteration $t+1$. We have

$$q_C(t) = \sum_{j=n-k+1}^{n-1} \binom{n-t}{j-t} 2^{-(n-t)}.$$

Note that $q_C(t) \leq 1/2$ for $t \leq n-2k-4$ as it is simply the normalized sum of some row of the Pascal's triangle starting after the mid point to the end. The random variable L_C follows the binomial law with $\lambda - l_A$ trials and success probability $(q_C(t)/(1-q_A(t)))$ as its conditional distribution. Therefore, the expected value of L_C is at most $0.58(\lambda - l_A)$. If $\delta = 0.125$ and $l_A < \lambda/4$, then $(3\lambda/4) - l_A > (1+\delta)0.58(\lambda - l_A)$.

For convenience, let $\mu^+ = 0.58(\lambda - l_A)$, and D represent the event $L_A < \lambda/4$. We have

$$\begin{aligned}\Pr\left[L_C < \frac{3\lambda}{4} - l_A \middle| D\right] &> \Pr\left[L_C < (1+\delta)\mu^+ \mid D\right] \\ &= 1 - \Pr\left[L_C \geq (1+\delta)\mu^+ \mid D\right] \\ &\geq 1 - \exp\left(-\frac{\lambda}{442}\right)\end{aligned}$$

via Lemma 2.

Let E be the event that the cgEA sets the selected position's marginal to 1. We have

$$\begin{aligned}\Pr[E] &\geq \Pr\left[L_C < \frac{3\lambda}{4} - l_A \middle| L_A < \frac{\lambda}{4}\right] \cdot \Pr\left[L_A < \frac{\lambda}{4}\right] \\ &\geq 1 - 2\exp\left(-\frac{\lambda}{442}\right).\end{aligned}$$

Hence, the probability that it does so for the first $n-2k-4$ iterations is at least

$$1 - 2n\exp\left(-\frac{\lambda}{442}\right)$$

via Bernoulli's inequality.

The probability of the algorithm sampling the optimum in iteration $n-2k-3$ if no marginal is at 0 is

$$1 - \left(1 - 2^{-(2k+4)}\right)^{\lambda} \geq 1 - \exp\left(-\frac{\lambda}{2^{2k+4}}\right).$$

Therefore, the probability of the cgEA optimizing the $\textsc{Jump}_k$ function is, via Lemma 3, at least

$$1 - \exp\left(-\frac{\lambda}{2^{2k+4}}\right) - 2n\exp\left(-\frac{\lambda}{442}\right).$$

If $\lambda = \Omega\left(4^k \ln n\right)$, the probability is at least $1 - n^{-\Omega(1)}$. ■

E. Analysis on JumpOffsetSpike

Theorem 5: Let $c > 0$ be a sufficiently large constant. For an arbitrary gap size $k \leq n/4$, the cgEA with population size $\lambda > c\,ek\sqrt{n}\ln n$ and a uniform starting distribution optimizes the n-dimensional $\textsc{JumpOffsetSpike}_k$ function with probability $1 - n^{-\Omega(1)}$ in at most $2n\lambda$ function evaluations. Furthermore, if $k < \sqrt{n} - 4$, then a population size $\lambda > c\sqrt{n}\ln n$ is sufficient.

Proof: The fitness valley in $\textsc{JumpOffsetSpike}_k$ is a subset of that in Jump with a gap size of $n/4$. We can, therefore, reuse the analysis of the Jump function and state that the cgEA will not set any marginal to 0 for the first $n/2-4$ iterations with probability at least $1 - 2n\exp(-(\lambda/442))$.

Denote by D the event that no marginal is at 0 after the first $n/2-4$ iterations. We condition on this event. Let q be the probability of sampling, uniformly at random, a bitstring of length $n/2+4$ with hamming weight $n/4 + k/2 + 4$. We have

$$q = \binom{\frac{n}{2}+4}{\frac{n}{4}+\frac{k}{2}+4} 2^{-(\frac{n}{2}+4)}.$$

For convenience, let $r = n/4+2$ and $s = k/2+2$. We apply Lemma 4 to q to obtain

$$q = \binom{2r}{r+s} 4^{-r} \geq \frac{\left(1-\frac{s}{r}\right)^s}{\sqrt{4r}}.$$

Let E denote the event that the cgEA optimizes the JOS_k function. We have

$$\begin{aligned}\Pr[E \mid D] &= 1 - (1-q)^{\lambda} \geq 1 - \exp(-\lambda q) \\ &\geq 1 - \exp\left(-\frac{\lambda\left(1-\frac{s}{r}\right)^s}{\sqrt{4r}}\right).\end{aligned}$$

We apply Lemma 3 to obtain

$$\Pr[E] \geq 1 - \exp\left(-\frac{\lambda\left(1-\frac{s}{r}\right)^s}{\sqrt{4r}}\right) - 4n\exp\left(-\frac{\lambda}{442}\right).$$

Since $k \leq n/4$, we have $(1-(s/r))^s \geq \exp(-2s)$ and

$$\Pr[E] \geq 1 - \exp\left(-\frac{\lambda e^{-(k+4)}}{\sqrt{n+8}}\right) - 4n\exp\left(-\frac{\lambda}{442}\right).$$

Algorithm 2: λ-*free* cgEA With Starting Distribution p to Maximize $f\colon \{0,1\}^n \to \mathbb{R}$

$r \leftarrow 0$ // round counter
Let $z^* \in \{0,1\}^n$ be drawn uniformly at random.
repeat
  $r \leftarrow r+1$
  Run the cgEA (Algorithm 1) to maximize f with population size 2^r and starting distribution p. Let z be its output.
  if $f(z) \geq f(z^*)$ **then** $z^* \leftarrow z$
until *termination criterion is met*
return z^*

If $\lambda = \Omega\left(e^k \sqrt{n} \ln n\right)$, the probability is at least $1 - n^{-\Omega(1)}$.

However, if $k = a(\sqrt{n} - 4)$ for $a \in [0,1]$, then $(1-(s/r))^s > 1 - a^2$ and

$$\Pr[E] \geq 1 - \exp\left(-\frac{\lambda\left(1-a^2\right)}{\sqrt{n+8}}\right) - 4n\exp\left(-\frac{\lambda}{442}\right).$$

If $\lambda = \Omega\left(\sqrt{n}\ln n\right)$, the probability is at least $1 - n^{-\Omega(1)}$. ■

V. λ-Free Competing Genes Evolutionary Algorithm

From the runtime analyses in the previous section, we see that the population size plays a crucial role in the success or failure of the algorithm. A simple restart strategy (Algorithm 2) that doubles the population size with each restart frees the user from specifying the population size upfront.

We now prove the following theorem about this restart strategy.

Theorem 6: Let λ^* be the minimum population size such that for every $\lambda' \in [\lambda^*, 2\lambda^*]$, the cgEA with population size λ' and a starting distribution p optimizes a pseudo-Boolean function $f\colon \{0,1\}^n \to \mathbb{R}$ with probability at least q. Then, the λ-*free* cgEA (Algorithm 2) optimizes f with probability at least q in at most $8n\lambda^*$ function evaluations.

Proof: Let $t = \lceil \log_2 \lambda^* \rceil$. At round t, the call to the cgEA optimizes f with probability at least q. The number of function evaluations accumulated from round 1 to round t is at most T, where

$$\begin{aligned} T &\leq 2n\sum_{r=1}^{t} 2^r = 2n\left(2^{t+1} - 2\right) \\ &< 2n\left(2^{2+\log_2 \lambda^*} - 2\right) \\ &< 8n\lambda^*. \end{aligned}$$

■

VI. Partial-Restart Competing Genes Evolutionary Algorithm

cgEA is a greedy algorithm that, in each iteration, assigns the bit value it considers more beneficial to the selected position. Like any other greedy algorithm, cgEA can make a wrong assignment, based on specious evidence, that prevents it from finding an optimal solution. Indeed, for any selected position, cgEA may pick the wrong value due to deception or when the sampled solutions are on a fitness plateau.

Algorithm 3: *partial-restart* cgEA With Population Size $\lambda \in \mathbb{N}, \lambda \geq 2$ and Reset Probability $\alpha \in [0,1]$ to Maximize $f\colon \{0,1\}^n \to \mathbb{R}$

for $i \in \{1,\ldots,n\}$ **do** $p_i \leftarrow 1/2$
Let $z^* \in \{0,1\}^n$ be drawn uniformly at random.
repeat
  Run the cgEA (Algorithm 1) to maximize f with population size λ and starting distribution p. Let z be its output.
  if $f(z) \geq f(z^*)$ **then** $z^* \leftarrow z$
  for $i \in \{1,\ldots,n\}$ **do**
    $p_i \leftarrow \begin{cases} 1/2 & \text{with probability } \alpha \\ z_i^* & \text{otherwise.} \end{cases}$
until *termination criterion is met*
return z^*

The order of position selection also presents another point of failure for the algorithm. For example, consider the following fitness function:

$$f(x) = \begin{cases} n+1, & \text{if } x_n = 0 \wedge x_i = 1 \ \forall i \in [n-1] \\ \|x\|_1, & \text{otherwise} \end{cases}$$

which is similar to the OneMax function, except its global optimum differs from the all-ones string at the last position. With high probability, the cgEA chooses and sets the last position to 1 long before the model produces solutions close to the optimum. The algorithm fails to find the new optimum string even though it is only one bitflip away from the old one. Different runs of the algorithm on this problem will produce different ordering in position selection, indicating the algorithm's difficulty in uncovering the structure of variable dependency.

In addition, complex variable interactions challenge univariate EDAs by manifesting as local optima and plateaus that trap algorithms in subpar regions of the fitness landscape. The algorithm's failure on this modified OneMax function also highlights that the local search phase of the cgEA is to help find the best value for the selected position; it is not a local search around the best-so-far solution. So while the cgEA aims to avoid genetic drift, it can still fail to optimize a fitness function due to epistasis.

To remedy these issues, we augment the cgEA with the ability to search the neighborhood of the best-so-far solution by revisiting prior assignments. We call this variant the *partial-restart* cgEA (Algorithm 3). The restarts are partial because we bias the starting distribution toward the best-so-far solution while resetting only a fraction of the marginals. The *partial-restart* cgEA introduces a parameter $\alpha \in [0,1]$ as the probability of resetting each marginal to the uniform distribution in the next round.

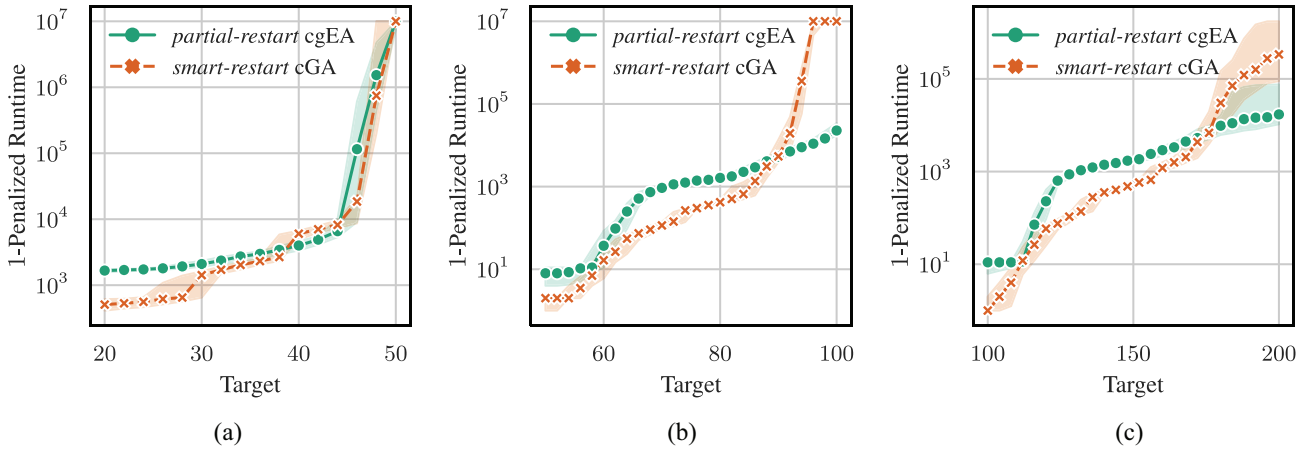

Fig. 1. Semi-logarithmic plots of the 1-Penalized runtime of the *partial-restart* cgEA and the *smart-restart* cGA to find a solution whose fitness is at least a specified target value for 100-dimensional instances of three benchmark problems. The dashed lines depict the medians, while the bands show the range of values between the 25th and 75th percentiles. Each plot is over 100 independent runs. (a) MIVS. (b) IsingRing. (c) IsingTorus.

VII. Experiments

To demonstrate the effectiveness of the *partial-restart* cgEA at escaping local optima, we empirically investigated the 1-Penalized Runtime of the *partial-restart* cgEA on the MIVS, IsingRing and IsingTorus problems. We included the *smart-restart* cGA [21] as a baseline. The c-Penalized Runtime of an algorithm A to meet a target value v when optimizing a fitness function f in a run is defined as

$$\text{c-Penalized Runtime}(v) = \min\{cB,\ T(A,B,f,v)\}$$

where $T(A,B,f,v) \in \mathbb{N} \cup \{\infty\}$ is the number of function evaluations out of a budget of B evaluations that algorithm A needed in that run to generate a solution whose fitness is at least v. If algorithm A does not generate such a solution, then $T(A,B,f,v) = \infty$.

Fig. 1 summarizes the empirical distribution of the 1-Penalized Runtime over 100 independent runs of the *partial-restart* cgEA and *smart-restart* cGA on 100-dimensional instances of the MIVS, IsingRing and IsingTorus problems. Each run had a budget of 10^7 function evaluations. For the *smart-restart* cGA, we used a budget factor of 8 and an update factor of 2. For the *partial-restart* cgEA, we used a population size λ of 10 and a reset probability α of 0.1.

All three instances each have two global optima, and for each instance, the best bit value in any position depends on the values at neighboring positions. The itinerant *partial-restart* cgEA makes locally beneficial assignments in the positions it visits, thereby creating solutions that appear to be splices of the two optimal strings. The partial restarts allow a reconstruction of portions of the best-so-far solution, pushing it closer to one of the optima. The *partial-restart* cgEA's manner of search does not confer any advantage on the MIVS problem, which has several almost-optimal strings that are far away from either optima string.

A phenomenon common to the plots is the *partial-restart* cgEA's weak performance in the early stages of the search process. This phenomenon results from the delay in exploiting the best solution the algorithm has found until the next restart. The code for these experiments is available on GitHub.[1]

VIII. Conclusion

We introduced a new univariate EDA (cgEA) to avoid genetic drift and analyzed its runtime on five benchmark functions. We proved that the cgEA optimizes any Monotone function in $O(n)$ evaluations, the LeadingOnes and DeceptiveLeadingBlocks functions in $O(n \ln n)$ evaluations, the Jump_k function in $O\left(4^k n \ln n\right)$ evaluations, and the JumpOffsetSpike_k function in $O\left(e^k n^{1.5} \ln n\right)$ evaluations for optimal setting of the population size. The good runtimes on these functions result from how the algorithm updates its probability model in a fashion akin to coordinate descent. The λ-*free* cgEA extends the algorithm by removing the need for the user to specify the population size.

So far, our treatment has assumed that fitness evaluations are reliable, but real-world optimization problems may have evaluations corrupted by noise. One way to handle this issue is by making incremental updates to the marginals. The increments may be fixed-size like the cGA's, or it can factor in the difference in votes.

Although we introduced the *partial-restart* cgEA to escape local maxima caused by complex interactions of the decision variables, we can also extend the cgEA to instead model such complex dependencies by updating multiple marginals in one iteration. There are strategies from Block Coordinate Descent methods [38] that specify how to find a group of variables to update. In particular, Tseng and Yun [39], selected the variables corresponding to the gradient elements within a multiplicative factor of being maximum in magnitude. However, capturing complex dependencies usually results in increased model complexity that requires more time and data to learn. For example, the local search phase of the cgEA will be exponentially expensive in the number of selected variables.

Despite all these modifications, it is easy to create problems that the modified algorithm will find hard to optimize. Suppose

[1] https://github.com/adetunji-david/cgea-experiment

we have a problem p that the cgEA or its extensions optimize efficiently. We can create a new problem p' that is similar to p but challenging to solve by moving its global optimum into a region of the search space that the algorithm rarely visits. Even though global optimization is elusive, we still expect these modifications or combinations to generate good-enough solutions for practical optimization problems.

REFERENCES

[1] 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.

[2] H. Mühlenbein and G. Paaß, "From recombination of genes to the estimation of distributions I. Binary parameters," in *Proc. Int. Conf. Parallel Problem Solving Nat.*, 1996, pp. 178–187.

[3] S. Baluja, "Population-based incremental learning. A method for integrating genetic search based function optimization and competitive learning," Dept. Comput. Sci., Carnegie-Mellon Univ., Pittsburgh, PA, USA, Rep. CMU-CS-94-163, 1994.

[4] F. Neumann, D. Sudholt, and C. Witt, "A few ants are enough: ACO with iteration-best update," in *Proc. 12th Annu. Conf. Genet. Evol. Comput.*, 2010, pp. 63–70.

[5] M. Pelikan, M. W. Hauschild, and F. G. Lobo, "Estimation of distribution algorithms," in *Springer Handbook of Computational Intelligence*. Berlin, Germany: Springer, 2015, pp. 899–928.

[6] B. Doerr and W. Zheng, "Sharp bounds for genetic drift in estimation of distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1140–1149, Dec. 2020.

[7] M. S. Krejca, "Theoretical analyses of univariate estimation-of-distribution algorithms," Ph.D. dissertation, Dept. Digit. Eng., Universität Potsdam, Brandenburg, Germany, 2019.

[8] T. Friedrich, T. Kötzing, and M. S. Krejca, "EDAs cannot be balanced and stable," in *Proc. Genet. Evol. Comput. Conf.*, 2016, pp. 1139–1146.

[9] J. L. Shapiro, "Drift and scaling in estimation of distribution algorithms," *Evol. Comput.*, vol. 13, no. 1, pp. 99–123, Mar. 2005.

[10] B. Doerr and M. S. Krejca, "The univariate marginal distribution algorithm copes well with deception and epistasis," *Evol. Comput.*, vol. 29, no. 4, pp. 543–563, Dec. 2021.

[11] P. K. Lehre and P. T. H. Nguyen, "On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help," in *Proc. 15th ACM/SIGEVO Conf. Found. Genet. Algorithms*, 2019, pp. 154–168.

[12] 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.

[13] 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.

[14] M. S. Krejca and C. Witt, "Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax," *Theor. Comput. Sci.*, vol. 832, pp. 143–165, Sep. 2020.

[15] 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.

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

[17] J. Lengler, D. Sudholt, and C. Witt, "The complex parameter landscape of the compact genetic algorithm," *Algorithmica*, vol. 83, no. 4, pp. 1096–1137, 2021.

[18] B. Doerr and C. Doerr, "Theory of parameter control for discrete black-box optimization: Provable performance gains through dynamic parameter choices," in *Theory of Evolutionary Computation*. Cham, Switzerland: Springer Int., 2020, pp. 271–321.

[19] M. Pelikan and T.-K. Lin, "Parameter-less hierarchical BOA," in *Proc. Genet. Evol. Comput. Conf.*, 2004, pp. 24–35.

[20] B. Doerr, "The runtime of the compact genetic algorithm on jump functions," *Algorithmica*, vol. 83, no. 10, pp. 3059–3107, 2021.

[21] B. Doerr and W. Zheng, "From understanding genetic drift to a smart-restart parameter-less compact genetic algorithm," in *Proc. Genet. Evol. Comput. Conf.*, 2020, pp. 805–813.

[22] H. Mühlenbein, "The equation for response to selection and its use for prediction," *Evol. Comput.*, vol. 5, no. 3, pp. 303–346, Sep. 1997.

[23] B. Doerr and M. S. Krejca, "Significance-based estimation-of-distribution algorithms," *IEEE Trans. Evol. Comput.*, vol. 24, no. 6, pp. 1025–1034, Dec. 2020.

[24] J. E. Rowe and Aishwaryaprajna, "The benefits and limitations of voting mechanisms in evolutionary optimisation," in *Proc. 15th ACM/SIGEVO Conf. Found. Genet. Algorithms*, 2019, pp. 34–42.

[25] M. S. Krejca and C. Witt, "Theory of estimation-of-distribution algorithms," in *Theory of Evolutionary Computation*. Cham, Switzerland: Springer, 2020, pp. 405–442.

[26] A. Auger and N. Hansen, "A restart CMA evolution strategy with increasing population size," in *Proc. IEEE Congr. Evol. Comput.*, vol. 2, 2005, pp. 1769–1776.

[27] C. Witt, "On crossing fitness valleys with majority-vote crossover and estimation-of-distribution algorithms," in *Proc. 16th ACM/SIGEVO Conf. Found. Genet. Algorithms*, 2021, pp. 1–15.

[28] C. Doerr, F. Ye, N. Horesh, H. Wang, O. M. Shir, and T. Bäck, "Benchmarking discrete optimization heuristics with IOHprofiler," *Appl. Soft Comput.*, vol. 88, Mar. 2020, Art. no. 106027.

[29] W. Hoeffding, "Probability inequalities for sums of bounded random variables," *J. Amer. Stat. Assoc.*, vol. 58, no. 301, pp. 13–30, 1963.

[30] B. Doerr, "Probabilistic tools for the analysis of randomized optimization heuristics," in *Theory of Evolutionary Computation*. Cham, Switzerland: Springer Int., 2020, pp. 1–87.

[31] Z. Sasvári, "Inequalities for binomial coefficients," *J. Math. Anal. Appl.*, vol. 236, no. 1, pp. 223–226, 1999.

[32] S.-I. Amari, "Natural gradient works efficiently in learning," *Neural Comput.*, vol. 10, no. 2, pp. 251–276, 1998.

[33] J. Martens, "New insights and perspectives on the natural gradient method," *J. Mach. Learn. Res.*, vol. 21, no. 1, pp. 1–76, Jan. 2020.

[34] S.-I. Amari and S. C. Douglas, "Why natural gradient?" in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, vol. 2, 1998, pp. 1213–1216.

[35] D. Wierstra, T. Schaul, T. Glasmachers, Y. Sun, J. Peters, and J. Schmidhuber, "Natural evolution strategies," *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 949–980, 2014.

[36] Y. Ollivier, L. Arnold, A. Auger, and N. Hansen, "Information-geometric optimization algorithms: A unifying picture via invariance principles," *J. Mach. Learn. Res.*, vol. 18, no. 18, pp. 1–65, 2017.

[37] D. G. Luenberger and Y. Ye, *Linear and Nonlinear Programming*, 3rd ed. New York, NY, USA: Springer, 2008, p. 253.

[38] J. Nutini, I. Laradji, and M. Schmidt, "Let's make block coordinate descent converge faster: Faster greedy rules, message-passing, active-set complexity, and Superlinear convergence," *J. Mach. Learn. Res.*, vol. 23, no. 131, pp. 1–74, 2022.

[39] P. Tseng and S. Yun, "A coordinate gradient descent method for non-smooth separable minimization," *Math. Program.*, vol. 117, nos. 1–2, pp. 387–423, 2009.

Adetunji David Ajimakin is currently pursuing the Ph.D. degree with the Indian Institute of Science, Bengaluru, India.

His research interests lie in the intersection of machine learning and evolutionary computation.

V. Susheela Devi received the Ph.D. degree from the Indian Institute of Science, Bengaluru, India, in 2000.

She currently serves as a Principal Research Scientist with the Indian Institute of Science. She works in the areas of machine learning, deep learning, artificial intelligence, and soft computing.