# General Univariate
# Estimation-of-Distribution Algorithms

Benjamin Doerr[1(✉)] and Marc Dufay[2]

[1] LIX, CNRS, École Polytechnique, Institut Polytechnique de Paris,
Palaiseau, France
`doerr@lix.polytechnique.fr`
[2] École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

**Abstract.** We propose a general formulation of a univariate estimation-of-distribution algorithm (EDA). It naturally incorporates the three classic univariate EDAs *compact genetic algorithm*, *univariate marginal distribution algorithm* and *population-based incremental learning* as well as the *max-min ant system* with iteration-best update. Our unified description of the existing algorithms allows a unified analysis of these; we demonstrate this by providing an analysis of genetic drift that immediately gives the existing results proven separately for the four algorithms named above. Our general model also includes EDAs that are more efficient than the existing ones and these may not be difficult to find as we demonstrate for the ONEMAX and LEADINGONES benchmarks.

**Keywords:** Estimation of distribution algorithms · Genetic drift ·
Running time analysis · Theory

## 1 Introduction

Estimation-of-distribution algorithms (EDAs) are a class of iterated randomized search heuristics proposed first in the 1990s [21]. Different from genetic algorithms (GAs), which evolve a set $P$ ("population") of good solutions for a given problem, EDAs evolve a probability distribution ("probabilistic model") on the set of possible solutions, hopefully in the way that good solutions have a higher probability assigned to them. Since it is clear that a set $P$ of solutions can be represented by a probability distribution (namely the uniform distribution on $P$), EDAs (with an appropriate probabilistic model) have a much richer way of transporting information from one iteration to the next than genetic algorithms.

Several results show that this theoretical advantage can be turned into a true advantage when running the EDA in the right way. For example, it was shown that the more cautious way of updating the probabilistic model of EDAs (as opposed to the only alternatives of a GA, which are to accept or discard a solution) can lead to a high robustness to noise [15,16]. The fact that EDAs can sample with a larger variance was shown to be advantageous for leaving local

optima [5,8,18,38]. In [7], it was demonstrated that the probabilistic model developed by an EDA allows to obtain much more diverse good solutions than what can be achieved by population-based algorithms.

Due to their higher simplicity, the most studied form of EDAs are *univariate* ones, which sample the variables of each solution independently. When restricting ourselves to pseudo-Boolean optimization, that is, the solutions are bit-strings of length $n$, then this means that the probabilistic model can be described by a *frequency vector* $p = (p_1, \ldots, p_n) \in [0,1]^n$ such that a sample $x \in \{0,1\}^n$ from this model satisfies

$$\Pr[x_i = 1] = p_i \text{ independently for all } i \in [1..n] := \{1, \ldots, n\}. \tag{1}$$

The three classic univariate EDAs are *population-based incremental learning (PBIL)* [2], the *univariate marginal distribution algorithm (UMDA)* [28], and the *compact genetic algorithm (cGA)* [17]. As observed in [22], the *max-min ant system (MMAS)* [35] with iteration-best pheromone update also is a univariate EDA (when used for pseudo-Boolean optimization). We note that the UMDA and this MMAS are special cases of PBIL. Unfortunately, with very few results existing for the PBIL, this connection so far could not be exploited extensively.

So far, these four algorithms have mostly been discussed separately, and for many aspects, only one or two of the four algorithms have been regarded. For example, there are only two mathematical analysis on how EDAs cope with Gaussian noise and these regards only the cGA [16] and the MMAS [15]. For the question how EDAs cope with local optima, the existing runtime analyses only regard the cGA [5,18,38] and the MMAS [3]. This leaves many questions unanswered.

We also note that many arguments used in the past were specific to the particular algorithm regarded. For example, the analyses in [5,18] exploit that the cGA enjoys the property that if the sample with better fitness is closer to the optimum, then the model update will reduce the expected distance of the samples from the optimum. The MMAS does not have this property and consequently, a different proof approach was necessary in [3].

**Our Results:** In this work, we try to improve this situation by proposing a simple, yet general class of EDAs that includes the four algorithms mentioned above. Our hope is that by thus distilling the common features of these algorithms, it becomes easier to find analyses that apply simultaneously to all four algorithms. We demonstrate that this is indeed possible by proving a quantitative statement on the genetic drift effect in our EDA class. This result contains as special cases the results (separately) proven in [12].

Our second hope is that the large class of EDAs defined by our model also contains algorithms with better performance than the four known algorithms. With elementary non-rigorous arguments, we design such an EDA and show via an experimental analysis that it is at least twice as fast at the cGA and UMDA with optimized parameters on the OneMax benchmark. We note that this new algorithm is in no way more complicated than the known special cases of our general model – it just profits from wider ranges of allowed parameters.

## 2    Previous Work

For reasons of space and since several good surveys and textbooks are available, we describe here only the works that are really close to ours. For a general introduction to EDAs and details on applications, we refer to the surveys [19, 25, 31].

Our work, while not purely mathematical, nevertheless is regarding EDAs more from a theoretical perspective. A very recent survey on the state of the art of the theory of EDAs is [22], broader introductions to theoretical approaches in evolutionary computation include [1, 10, 20, 29]. As can easily be deduced from this survey, the theoretical understanding of EDAs is far from complete and for many basic questions, e.g., the runtime on the simple ONEMAX benchmark, a complete answer is still missing. What can also be observed from this survey is that essentially all previous works regard only a single univariate EDA. There are few exceptions, e.g., in [36] both the cGA and the MMAS is analyzed, but also in these cases the results for different algorithms are proven separately.

The only previous work we are aware of that undertakes an attempt towards a unified treatment of univariate EDAs is [14]. There, the framework of an $n$-Bernoulli-$\lambda$-EDA is defined. This framework is very general and includes not only our EDA model, but in fact all univariate EDAs which sample a fixed number $\lambda$ of offspring according to (1) and then update the probabilistic model $p$ via any deterministic function $\phi$ that takes as arguments the current model and the offspring together with their fitness. Not surprisingly, in such an extremely general model it is hard to prove meaningful results, and consequently, the particular results in [14] need non-trivial additional assumptions: To show that a stable EDA is not balanced, in particular the additional assumption is made that whenever the EDA optimizes a function with neutral $i$-th bit, then at all times $t$ the sampling frequency $p_i(t)$ satisfies $\mathrm{Var}[p_i(t+1) \mid p_i(t)] = -ap_i(t)^2 + bp_i(t) + c$ for suitable $a, b, c \in \mathbb{R}$ with $0 < a < 1$, see [14, Theorem 10] (this notion has been relaxed to the requirement that $\inf\{\mathrm{Var}[p_i(t+1) + \mathbf{1}[p_i(t) \notin [d, 1-d]] \mid p_i(t)] \mid t \in \mathbb{N}\} > 0$ for some $d = o(1)$ in [23, Theorem 6.11]). Similarly, the runtime analysis on the LEADINGONES benchmark relies on two specific assumptions how the frequencies behave during the optimization process [14, Theorem 12]. There is no doubt that also with these restrictions, the results in [14] are strong and impressive, but the need for the restrictions suggests that the $n$-Bernoulli-$\lambda$-EDA model is too general to admit strong results covering the whole model (and this is where we hope that our more narrow model is more effective).

There have also been some attempts to encompass EDAs in a model even wider. One of them is by defining these algorithms as *model-based search* algorithms which rely on a parameterized probabilistic model as opposed to *instance-based search* algorithms which rely on a population of solutions [39]. A model-based search algorithm is described by its probabilistic model and the way it updates its model and some parallels can be made between univariate EDAs and gradient-based methods. Another approach described in [30] is by turning existing EDAs into a continuous-time black-box optimization method using the *information-geometric optimization* (IGO) method which can then be turned

back into algorithms using time discretization. Existing univariate algorithms like cGA or PBIL can be retrieved using this method. However, these approaches result in a model that is too general to obtain running time results or to obtain ideas how to set the parameter of the algorithms.

## 3    Univariate EDA: Classic and New

In this section, we first describe briefly the four existing algorithms mentioned in the introduction and then derive from these a general model encompassing all four. We shall write $x \sim \text{Sample}(p)$ to denote that $x \in \{0,1\}^n$ is sampled according to the univariate model described by the frequency vector $p \in [0,1]^n$, that is, that $x$ satisfies (1). We assume that each call of this sampling procedure is stochastically independent from all other samplings and possibly other random decisions of the algorithm. When an algorithm optimizing a function $f$ samples $\lambda$ individuals, we denote these by $x[1], \ldots, x[\lambda]$ and we denote by $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$ the sorting of these by decreasing (worsening) fitness $f$, with ties broken randomly. All algorithms initialize the univariate model as $p = (\frac{1}{2}, \ldots, \frac{1}{2})$, which gives the uniform distribution on the search space $\{0,1\}^n$. In their main loop, all sample a certain number of solutions und update the model based on the fitness of the solutions. We first describe all algorithms in the basic version without artificial frequency margins, then propose our general EDA model (also without frequency margins), and finally discuss how to include such margins.

The *compact genetic algorithm (cGA)* [17] samples only two solutions and modifies the frequency vector by a scalar multiple of the difference between the better and the worse solution, that is, $p \leftarrow p + \frac{1}{K}(\tilde{x}[1] - \tilde{x}[2])$. Here $K$ is the only algorithm parameter called *hypothetical population size*. In other words, a frequency $p_i$ does not change if the two samples agree in the $i$-th bit, and it moves by an additive term of $\frac{1}{K}$ towards the bit value of the better solution otherwise. Usually, $K$ is taken as an even integer since this automatically keeps the frequencies in the range $[0,1]$. For other values of $K$, one would need to cap the frequencies after the update into the interval $[0,1]$.

The *univariate marginal distribution algorithm (UMDA)* [28] with parameters $\lambda, \mu \in \mathbb{Z}_{\geq 1}$ samples $\lambda$ solutions and updates the model to the average of the $\mu$ best solutions, that is, $p \leftarrow \frac{1}{\mu} \sum_{i=1}^{\mu} \tilde{x}[i]$.

The *max-min ant system (MMAS)* [35] with iteration-best update besides the sample size $\lambda$ has the *learning rate* $\rho \in ]0,1]$ (*pheromone evaporation rate* in the ant colony optimization language) as second parameter. Only the best offspring is used for the model update and it enters the model with weight $\rho$, that is, the model update is $p \leftarrow (1 - \rho)p + \rho\tilde{x}[1]$.

*Population-based incremental learning (PBIL)* [2] selects $\mu$ out of $\lambda$ solutions and combines their average weighted by $\rho$ with the current model: $p \leftarrow (1 - \rho)p + \rho\frac{1}{\mu} \sum_{i=1}^{\mu} \tilde{x}[i]$. Consequently, PBIL has as special cases both the UMDA (by taking $\rho = 1$) and the MMAS (by taking $\mu = 1$).

The pseudocodes for these four algorithms are given in Algorithms 1 to 4. As can easily be seen, in all four cases the new model is a linear combination of the

samples and the old model. This suggests the following *general univariate EDA model*. Let $\lambda \in \mathbb{Z}_{\geq 1}$ the sample size and $\gamma_0, \gamma_1, \ldots, \gamma_\lambda \in \mathbb{R}$ such that $\sum_{i=0}^{\lambda} \gamma_i = 1$. The general univariate EDA in its main loop samples $\lambda$ solutions and updates the frequency vector to $p \leftarrow \gamma_0 p + \sum_{i=1}^{\lambda} \gamma_i \tilde{x}[i]$, where this is to be understood that frequencies below zero or above one are replaced by zero or one. The complete pseudocode is given in Algorithm 5.

---

**Algorithm 1:** The cGA with parameter $K > 0$, maximizing a given function $f : \{0,1\}^n \to \mathbb{R}$.

---

**1** $p(0) = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right) \in [0,1]^n$
**2 for** $t = 1, 2, \ldots$ **do**
**3**     $x[1] \sim \text{Sample}(p(t-1))$
**4**     $x[2] \sim \text{Sample}(p(t-1))$
**5**     **if** $f(x[1]) \geq f(x[2])$ **then**
**6**         $p(t) = p(t-1) + \frac{1}{K}(x[1] - x[2])$
**7**     **else**
**8**         $p(t) = p(t-1) + \frac{1}{K}(x[2] - x[1])$
**9**     $p(t) = \max(0, \min(1, p(t)))$

---

**Algorithm 2:** The UMDA with parameters $\lambda \in \mathbb{Z}_{\geq 1}$ and $\mu \in [1..\lambda]$.

---

**1** $p(0) = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right) \in [0,1]^n$
**2 for** $t = 1, 2, \ldots$ **do**
**3**     **for** $i = 1, 2, \ldots, \lambda$ **do**
**4**         $x[i] \sim \text{Sample}(p(t-1))$
**5**     Sort the individuals into $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$ ordered by worsening fitness
**6**     *%% Update the frequency*
**7**     $p(t) = \frac{1}{\mu} \sum_{i=1}^{\mu} \tilde{x}[i]$

---

We immediately see that the general univariate EDA contains the four algorithms above as special cases. We obtain the cGA by taking $\lambda = 2$, $\gamma_0 = 1$, $\gamma_1 = \frac{1}{K}$, and $\gamma_2 = -\frac{1}{K}$. For the UMDA with parameters $\lambda$ and $\mu$, we use the same $\lambda$ and the weights $\gamma_0 = 0$, $\gamma_1 = \cdots = \gamma_\mu = \frac{1}{\mu}$ and $\gamma_{\mu+1} = \cdots = \gamma_\lambda = 0$. The MMAS results from taking $\gamma_0 = 1 - \rho$, $\gamma_1 = \rho$, and $\gamma_2 = \cdots = \gamma_\lambda = 0$. Finally, PBIL is the general EDA with $\gamma_0 = 1 - \rho$, $\gamma_1 = \cdots = \gamma_\mu = \frac{\rho}{\mu}$, and $\gamma_{\mu+1} = \cdots = \gamma_\lambda = 0$.

---

**Algorithm 3:** The MMAS with parameters $\lambda \in \mathbb{Z}_{\geq 1}$ and evaporation factor $\rho \in \,]0,1]$.

---

1   $p(0) = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right) \in [0,1]^n$
2   **for** $t = 1, 2, \ldots$ **do**
3       **for** $i = 1, 2, \ldots, \lambda$ **do**
4           $x[i] \sim \mathrm{Sample}(p(t-1))$
5       Find an individual with the best fitness $\tilde{x}[1]$
6       *%% Update the frequency*
7       $p(t) = (1-\rho)p(t-1) + \rho\tilde{x}[1]$

---

---

**Algorithm 4:** PBIL with parameters $\rho \in \,]0,1]$, $\lambda \in \mathbb{N}$ and $\mu \in [1..\lambda]$.

---

1   $p(0) = \left(\frac{1}{2}, \ldots, \frac{1}{2}\right) \in [0,1]^n$
2   **for** $t = 1, 2, \ldots$ **do**
3       **for** $i = 1, 2, \ldots, \lambda$ **do**
4           $x[i] \sim \mathrm{Sample}(p(t-1))$
5       Sort the individuals into $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$ ordered by their fitness
6       *%% Update the frequency*
7       $p(t) = (1-\rho)p(t-1) + \frac{\rho}{\mu} \sum_{i=1}^{\mu} \tilde{x}[i]$

---

## 4   Genetic Drift

Genetic drift is the phenomenon that the sampling frequencies of the probabilistic model move in some direction not because of the feedback from the fitness, but by an unfortunate accumulation of the small random movements that occur when there is no clear signal from the fitness. Genetic drift is problematic in that it can move frequencies close to the boundary values 0 and 1, where they tend to stay longer. This phenomenon and its drawbacks were first discussed in the series of works [32–34]. After a long sequence of fundamental results such as [4,12–14,24,27,36,37], mostly runtime analyses which only apply to a regime with low genetic drift, we now understand this phenomenon quite well. For reasons of completeness, we note that EDAs can also be successful in regimes with genetic drift, see, e.g., the runtimes results [4,37] for the UMDA on ONEMAX and LEADINGONES when the population size is logarithmic, but the general understanding is that genetic drift is dangerous and examples like the analyses of the UMDA on the DLB problem [9,26] show that genetic drift can lead to drastic performance losses.

The tightest quantitative statements on genetic drift were given in [12]. They were proven via separate analyses for the cGA and PBIL (which imply the corresponding results for the UMDA and MMAS). With our general model for univariate EDAs, we can now provide a unified analysis for these classic algorithms (and all algorithms that will be defined in the future that fit into this model).

---

**Algorithm 5:** Our general EDA algorithm defined by $(\gamma_i)_{i=0,\ldots,n}$ such that $\sum_{i=0}^{\lambda} \gamma_i = 1$.

---

**1** $p(0) = \left(\frac{1}{2},\ldots,\frac{1}{2}\right) \in [0,1]^n$
**2 for** $t = 1, 2, \ldots$ **do**
**3**  $\quad$ %%Sample the individuals
**4**  $\quad$ **for** $i = 1, 2, \ldots, \lambda$ **do**
**5**  $\quad\quad$ %%Generate the i-th individual $x[i]$
**6**  $\quad\quad$ $x_t[i] \sim \text{Sample}(p(t-1))$
**7**  $\quad$ Sort the individuals into $\tilde{x}_t[1], \ldots, \tilde{x}_t[\lambda]$ by worsening fitness
**8**  $\quad$ %% Update the frequency
**9**  $\quad$ $p(t) = \max(0, \min(1, \gamma_0 p(t-1) + \sum_{i=1}^{\lambda} \gamma_i \tilde{x}[i]))$

---

Genetic drift is usually studied by regarding a neutral bit, that is, a bit that has no influence on the fitness (note that such results imply similar results for bits that are neutral only for a certain time as in the LEADINGONES benchmark or bits that have a preference for one value as in monotonic functions, see [12]). By symmetry, the expected value of the sampling frequency of a neutral bit is always $\frac{1}{2}$ (and in fact, the distribution of this frequency is also symmetric around $\frac{1}{2}$). Nevertheless, as discussed above, the random fluctuations stemming from the updates of the probabilistic model will move this frequency towards the boundary values 0 and 1, and this is the phenomenon of genetic drift. Genetic drift can be quantified, e.g., via statements on the first time that the frequency leaves some middle ground, e.g., the interval $[\frac{1}{3}, \frac{2}{3}]$.

In the remainder of this section, let us assume that the first bit of our objective function $f$ is neutral. Then this bit has no influence on the selection, and consequently for all $i \in [1..\lambda]$, we have $\tilde{x}_1[i] \sim \mathcal{B}(p_1(t-1))$. For simplicity, we write $x_t^i = \tilde{x}_1[i], p_t = p_1(t)$ for all $t \geq 0, i \in [1..\lambda]$. We will also assume that we are not in a totally degenerate case, so there exists $i \in [1..\lambda]$ such that $\gamma_i \neq 0$.

**Lemma 1.** *The sequence* $\left(\frac{p_t(1-p_t)}{(1-\sum_{i=1}^{\lambda} \gamma_i^2)^t}\right)_{t \geq 0}$ *with respect to the filtration* $(p_t)_{t \geq 0}$ *is a martingale.*

We note that this result is quite beautiful because it gives a good insight on the behavior of a neutral bit and no approximation was needed, allowing us to obtain a martingale and not a supermartingale or a submartingale like what is usually the case. For reasons of space, the formal proof of this and the other results of this paper had to be omitted. They can be found in the appendix of the preprint [6].

Using this result, we can find an upper bound on the expected time for a neutral bit frequency to move away from 1/2.

**Lemma 2.** *Let* $T_L = \min\{t \geq 0, p_t \leq 1/3 \text{ or } p_t \geq 2/3\}$ *be the first time for a neutral bit to leave* $[1/3, 2/3]$. *Then* $E[T_L] = \mathcal{O}\left(\frac{1}{\sum_{i=1}^{\lambda} \gamma_i^2}\right)$.

To obtain a lower bound and more precise concentration results, we can use a Hoeffding inequality in a way similar, but more general than what was done in [12].

**Lemma 3.** *For all $T \in \mathbb{N}$ and $\delta > 0$, we have*

$$P\left[\forall t \in [0..T], |p_t - 1/2| < \delta\right] \geq 1 - 2\exp\left(\frac{-\delta^2}{2\,T\sum_{i=1}^{\lambda}\gamma_i^2}\right).$$

With $T_0 = \frac{\left(\sum_{i=1}^{\lambda}\gamma_i^2\right)^{-1}}{4\cdot36\log n}$ and a union bound, we obtain the following guarantee that neutral frequencies stay away from the boundaries.

**Corollary 1.** *Assuming that all bits are independent and neutral, with high probability, before iteration $T_0$, all bits frequencies stay within the range $[1/3, 2/3]$.*

As in [12, part VI], this result can be extended to bits with a preference. For a fitness function $f$, we say that it is *weakly preferring* 1 in bit $i$ if for all $(x_1, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n) \in \{0,1\}^{n-1}$ we have

$$f(x_1, \ldots, x_{i-1}, 1, x_{i+1}, \ldots, x_n) \geq f(x_1, \ldots, x_{i-1}, 0, x_{i+1}, \ldots, x_n).$$

Many common fitness functions like ONEMAX or LEADINGONES are *weakly preferring* 1 in any bit.

**Corollary 2.** *If the fitness function is weakly preferring a 1 on all of its bits, then we have $P\left[\forall i \in [1..n], \forall t \in [0..T_0], p_i^t \geq 1/3\right] = 1 - o(1)$.*

## 5   Optimizing the $(\gamma_i)_i$

A second advantage of our general formulation of univariate EDAs, besides giving unified proofs, could be that this broad class of algorithms contains EDAs that are superior to the four special cases that have been regarded in the past. To help finding such algorithms, we now discuss the influence on the $\gamma_i$ on the optimization progress. Since different $\gamma_i$ might be profitable in different stages of the optimization progress, we analyze their effect in a single iteration, that is, we condition on the current frequency vector. To ease the notation, let us call this frequency vector $p$ (without any time index). Let $\tilde{x}[1], \ldots, \tilde{x}[\lambda]$ denote the $\lambda$ samples taking in this iteration, sorted already by decreasing fitness. Then, ignoring the influence of frequency boundaries, the next frequency vector $p'$ satisfies $p' = \gamma_0 p + \sum_{i=1}^{\lambda}\gamma_i\tilde{x}[i]$.

We would like to have an idea of what the optimal $(\gamma_i)$ with respect to minimizing the expected convergence time to reach the optimal solution would look like. To do so, we look during a single iteration for the ONEMAX function at the best distribution of $(\gamma_i)$ while keeping the genetic drift minimal. During iteration $t$, let $X(t)$ be a random variable following distribution $(p_i(t))_i$, we want

to maximize $E[f(X(t+1))]$ knowing the previous distribution. ONEMAX being linear, using the linearity of expectation on all the different bits, we have

$$E[f(X(t+1))] = \gamma_0 E[f(X(t))] + \sum_{i=1}^{\lambda} \gamma_i E[f(\tilde{x}[i])]$$

$$= \left(1 - \sum_{i=1}^{\lambda} \gamma_i\right) E[f(X(t))] + \sum_{i=1}^{\lambda} \gamma_i E[f(\tilde{x}[i])]$$

$$= E[f(X(t))] + \sum_{i=1}^{\lambda} \gamma_i \left(E[f(\tilde{x}[i])] - E[f(X(t))]\right).$$

Let us assume that $(\tilde{\gamma}_i)_i$ are optimal for the current iteration and let $\delta = \sum_{i=1}^{\lambda} \tilde{\gamma}_i^2$ be the genetic drift. Because this iteration maximizes the expected outcome of the next distribution while minimizing the genetic drift, it is a solution to

$$\text{Maximize: } E[f(X(t))] + \sum_{i=1}^{\lambda} \gamma_i \left(E[f(\tilde{x}[i])] - E[f(X(t))]\right)$$

$$\text{Subject to: } \sum_{i=1}^{\lambda} \gamma_i^2 \leq \delta$$

Both the function to optimize and the constraint are polynomial so differentiable. Moreover the set solution to the constraint is bounded and closed, so it is compact. Therefore an optimal solution exists and we can use the method of Lagrange multipliers to find it: there exists a Lagrange multiplier $\alpha \leq 0$ such that

$$\begin{bmatrix} E[f(\tilde{x}[1])] - E[f(X(t))] \\ E[f(\tilde{x}[2])] - E[f(X(t))] \\ \cdots \\ E[f(\tilde{x}[\lambda])] - E[f(X(t))] \end{bmatrix} + \alpha \begin{bmatrix} 2\tilde{\gamma}_1 \\ 2\tilde{\gamma}_2 \\ \cdots \\ 2\tilde{\gamma}_\lambda \end{bmatrix} = 0.$$

So $(\tilde{\gamma}_i)_i$ are proportional to $(E[f(\tilde{x}[i])] - E[f(X(t))])_i$. Because $(\tilde{x}[i])$ are sorted according to their fitness, $(E[f(\tilde{x}[i])])_i$ is decreasing so $(\tilde{\gamma}_i)_i$ should also be decreasing.

## 6    Designing New Univariate EDAs

In this section, we propose two new univariate EDAs (that is, EDAs within our framework with $\gamma_i$ that do not lead to one of the four classical algorithms) and analyze them via experimental means. Given the momentary state of the art in mathematical runtime analysis of EDAs, it seems out of reach to conduct a mathematical runtime analysis precise enough to make visible the influence of the $\gamma_i$ on the runtime. The main insight derived from this part of our work is that with not much effort, one can find univariate EDAs which outperform

the classic univariate EDAs. We conduct this line of research for the two classic benchmarks OneMax and LeadingOnes.

**OneMax:** Since univariate EDAs sample the bits independently and since in the OneMax benchmark each bit contributes the same to the fitness, we expect a somewhat regular behavior in a set of independent samples: Those with best fitness will have many bits set correctly, those with lowest fitness with miss many bit values. This, together with the considerations of the previous section, suggests to give more weights to better samples in the frequency update, and to do this in a somewhat continuous manner. One way of doing so is taking

$$\gamma_0 = 1 - \beta \sum_{i=1}^{\lambda} (1 - \tfrac{i}{\lambda/2}) \approx 1 \text{ and } \gamma_i = \beta(1 - \tfrac{i}{\lambda/2}) \text{ for } i \in [1..\lambda], \qquad (2)$$

where $\beta$ is a positive number still to be determined. While not perfectly symmetric, essentially here $\tilde{x}[i]$ and $\tilde{x}[\lambda - i]$ have weights of opposite sign, hence $\gamma_0$ is essentially one.

We compare this new EDA with the two classic ones UMDA and cGA with optimized parameters. We do not regard the other two classic EDAs since with their learning rate $\rho$ they are structurally quite different and it is less understood what are good parameter settings for these. We note that there is no indication in the literature that the MMAS or PBIL with their slightly cautious learning mechanism could outperform the other two algorithms on a simple unimodal benchmark such as OneMax.

For the UMDA and cGA, we determine good parameter values as follows. For the UMDA, we chose to fix $\lambda$ as $\lfloor \log n \sqrt{n} \rfloor$ since both theoretical and experimental results show that this leads to good performances [37]. We use the same value of $\lambda$ for our EDA. Still for the UMDA, we set $\mu = \lfloor \lambda/3 \rfloor$ as this gave the best expected runtimes in the experiments we conducted to opitmize the parameters of the UMDA. For cGA, the only parameters that needs to be determined is the hypthetical population size $K$. From [11, Fig. 1], we know that the expected runtime of the cGA on OneMax is roughly a unimodal function in $K$.[1] Since $\beta$ in our algorithm plays a similar role as $K$ in the cGA (namely it regulates the strength of the model update), we expect a similar unimodal dependence on $\beta$ for our algorithms, which we confirm in experiments. For that reason, for each problem size $n$ we determined the optimal values for $K$ and $\beta$ via ternary search.

Figure 1 displays the average (in 200 runs) runtime of these three algorithms for different problems sizes. These results show that our general algorithm with a gamma distribution that was not used in previous algorithms is about twice as fast as the optimized UMDA and cGA. This suggest that it is not too difficult to find in our broad class of univariate EDAs new algorithms which are significantly faster than the classic algorithms.

---

[1] We know that [27] proved that the runtime of the cGA on OneMax is not unimodal in $K$ when $n$ is large enough, but apparently this asymptotic results becomes relevant only for very large population sizes.
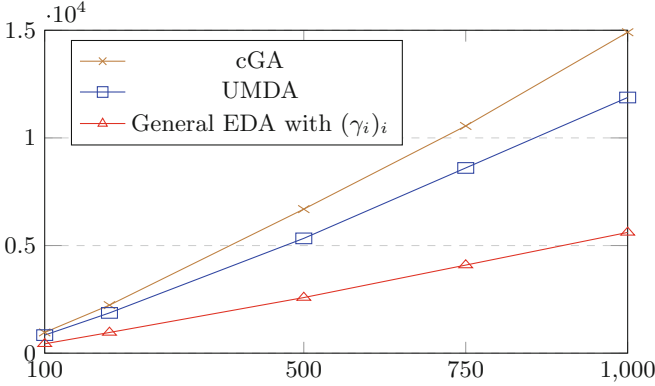
**Fig. 1.** Average running times (in fitness evaluations) of cGA (with optimized value of $K$), UMDA (with fixed $\lambda = \lfloor \log n\sqrt{n} \rfloor$ and optimized value $\mu = \lambda/3$), and our general algorithm with fixed gamma as in (2) and $\beta$ optimized, on the ONEMAX benchmark with problem size between $n = 100$ and $n = 1000$.

**LeadingOnes:** We undertook a similar work for the LEADINGONES benchmark. In this function, the bits do not contribute independently to the fitness, so our considerations valid in the design of the EDA above are not valid anymore. More detailedly, search points with low fitness reveal very little information how good solutions look like. For this reason, we design our new EDA in the way that such solutions are not taken into account for the model update. Without any optimizing, we set the cutoff for this regime at $\lambda/3$, that is, we have $\tilde{\gamma}_i = 0$ for all $i > \lambda/3$. For the remaining samples, we expect some positive information towards the optimum, and again we expect this to be stronger for better solutions, so we take $\tilde{\gamma}_i$ proportional to $\lfloor \lambda/3 \rfloor - (i - 1)$. With no particular reason, we decided to define an EDA resembling the UMDA, that is, we take $\tilde{\gamma}_0 = 0$ and

$$\tilde{\gamma}_i = \frac{\lfloor \lambda/3 \rfloor - (i - 1)}{\sum_{j=1}^{\lfloor \lambda/3 \rfloor} \lfloor \lambda/3 \rfloor - (j - 1)} \tag{3}$$

for all $i \in [1..\lambda/3]$.

In Fig. 2, we experimentally compare the EDA just designed, the EDA designed in the previous subsection, and the UMDA with parameters optimized (for LEADINGONES) as described in the previous subsection. As expected, the running time of our general algorithm with the $(\gamma_i)_i$ chosen in the previous subsection is not very good (roughly by 25% worse that the UMDA). The EDA just designed, however, beats the UMDA with optimized parameters by roughly 20%. This again shows that with moderately effort, one can find superior EDAs in the class of univariate EDAs defined in this work.

We admit that the ONEMAX and LEADINGONES benchmarks are well-understood, so designing a better univariate EDA for a complicated real-world problem will require more work. Nevertheless, we are optimistic that using

intuitive ideas such as the ones above, e.g., a continuous dependence of the $\gamma_i$ on the rank $i$, together with some trial-and-error experimentation can lead to good EDAs (better than the classic ones) also for more complex problems.
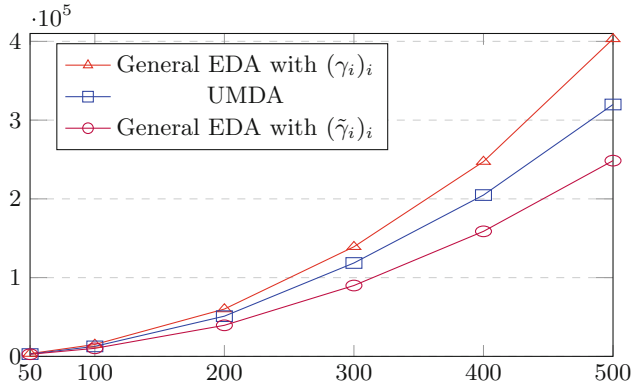


**Fig. 2.** Average running times (in fitness evaluations) over 200 runs of the classic UMDA (with optimized parameters) and the two EDAs designed in this section, on LEADINGONES with problem size between $n = 50$ and $n = 500$. The $\tilde{\gamma}_i$ chosen with consideration of elementary properties of LEADINGONES clearly outperform the other two algorithms.

## 7   Conclusion

In this work, we proposed a general formulation of a univariate EDA. It captures the three main univariate EDAs and the MMAS ant colony optimizer with iteration-best update. Our formulation allows to phrase proofs, so far conducted individually for the different algorithms, in a unified manner. We demonstrate this for a recent quantitative analysis of genetic drift. We are optimistic that our formulation also allows to conduct some of the existing runtime analyses in a unified manner. This would be particularly interesting as here many results have been shown only for some of the classic algorithms, e.g., the runtime analyses on the ONEMAX and JUMP benchmarks as well as the results on noisy optimization. However, given the high complexity of the existing analyses for particular algorithms, this might be a challenging task.

Our general formulation also allows to define new univariate EDAs, which might turn out to be superior to the existing ones. With intuitive arguments, we define such EDAs and show experimentally that they beat existing EDAs for the ONEMAX and LEADINGONES benchmarks. We are optimistic that this approach can be profitable also for other optimization problems.

# References

1. Auger, A., Doerr, B. (eds.): Theory of Randomized Search Heuristics. World Scientific Publishing (2011). https://doi.org/10.1142/7438

2. Baluja, S.: Population-based incremental learning: A method for integrating genetic search based function optimization and competitive learning. Tech. rep., Carnegie Mellon University (1994)

3. Benbaki, R., Benomar, Z., Doerr, B.: A rigorous runtime analysis of the 2-MMAS$_{ib}$ on jump functions: ant colony optimizers can cope well with local optima. In: Genetic and Evolutionary Computation Conference, GECCO 2021, pp. 4–13. ACM (2021). https://doi.org/10.1145/3449639.3459350

4. Dang, D.-C., Lehre, P.K., Nguyen, P.T.H.: Level-based analysis of the univariate marginal distribution algorithm. Algorithmica **81**(2), 668–702 (2018). https://doi.org/10.1007/s00453-018-0507-5

5. Doerr, B.: The runtime of the compact genetic algorithm on jump functions. Algorithmica **83**(10), 3059–3107 (2020). https://doi.org/10.1007/s00453-020-00780-w

6. Doerr, B., Dufay, M.: General univariate estimation-of-distribution algorithms (2022). CoRR abs/2206.11198

7. Doerr, B., Krejca, M.S.: Bivariate estimation-of-distribution algorithms can find an exponential number of optima. In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 796–804. ACM (2020). https://doi.org/10.1145/3377930.3390177

8. Doerr, B., Krejca, M.S.: A simplified run time analysis of the univariate marginal distribution algorithm on LeadingOnes. Theoret. Comput. Sci. **851**, 121–128 (2021). https://doi.org/10.1016/j.tcs.2020.11.028

9. Doerr, B., Krejca, M.S.: The univariate marginal distribution algorithm copes well with deception and epistasis. Evol. Comput. **29**, 543–563 (2021). https://doi.org/10.1162/evco_a_00293

10. Doerr, B., Neumann, F. (eds.): Theory of Evolutionary Computation-Recent Developments in Discrete Optimization. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4, http://www.lix.polytechnique.fr/Labo/Benjamin.Doerr/doerr_neumann_book.html

11. Doerr, B., Zheng, W.: From understanding genetic drift to a smart-restart parameter-less compact genetic algorithm. In: Genetic and Evolutionary Computation Conference, GECCO 2020, pp. 805–813. ACM (2020). https://doi.org/10.1145/3377930.3390163

12. Doerr, B., Zheng, W.: Sharp bounds for genetic drift in estimation-of-distribution algorithms. IEEE Trans. Evol. Comput. **24**, 1140–1149 (2020). https://doi.org/10.1109/TEVC.2020.2987361

13. Droste, S.: A rigorous analysis of the compact genetic algorithm for linear functions. Nat. Comput. **5**, 257–283 (2006). https://doi.org/10.1007/s11047-006-9001-0

14. Friedrich, T., Kötzing, T., Krejca, M.S.: EDAs cannot be balanced and stable. In: Genetic and Evolutionary Computation Conference, GECCO 2016, pp. 1139–1146. ACM (2016). https://doi.org/10.1145/2908812.2908895

15. Friedrich, T., Kötzing, T., Krejca, M.S., Sutton, A.M.: Robustness of ant colony optimization to noise. Evol. Comput. **24**, 237–254 (2016). https://doi.org/10.1162/EVCO_a_00178

16. Friedrich, T., Kötzing, T., Krejca, M.S., Sutton, A.M.: The compact genetic algorithm is efficient under extreme Gaussian noise. IEEE Trans. Evol. Comput. **21**, 477–490 (2017). https://doi.org/10.1109/TEVC.2016.2613739

17. Harik, G.R., Lobo, F.G., Goldberg, D.E.: The compact genetic algorithm. IEEE Trans. Evol. Comput. **3**, 287–297 (1999). https://doi.org/10.1109/4235.797971
18. Hasenöhrl, V., Sutton, A.M.: On the runtime dynamics of the compact genetic algorithm on jump functions. In: Genetic and Evolutionary Computation Conference, GECCO 2018, pp. 967–974. ACM (2018). https://doi.org/10.1145/3205455.3205608
19. Hauschild, M., Pelikan, M.: An introduction and survey of estimation of distribution algorithms. Swarm Evol. Comput. **1**, 111–128 (2011). https://doi.org/10.1016/j.swevo.2011.08.003
20. Jansen, T.: Analyzing Evolutionary Algorithms - The Computer Science Perspective. Springer (2013). https://doi.org/10.1007/978-3-642-17339-4
21. Juels, A., Baluja, S., Sinclair, A.: The equilibrium genetic algorithm and the role of crossover (1993), (Unpublished)
22. Krejca, M.S., Witt, C.: Theory of estimation-of-distribution algorithms. In: Theory of Evolutionary Computation. LNCS, pp. 405–442. Springer, Cham (2020). https://doi.org/10.1007/978-3-030-29414-4_9
23. Krejca, M.S.: Theoretical Analyses of Univariate Estimation-of-Distribution Algorithms. Ph.D. thesis, Universität Potsdam (2019)
24. Krejca, M.S., Witt, C.: Lower bounds on the run time of the univariate marginal distribution algorithm on OneMax. Theoret. Comput. Sci. **832**, 143–165 (2020). https://doi.org/10.1016/j.tcs.2018.06.004
25. Larrañaga, P., Lozano, J.A. (eds.): Estimation of Distribution Algorithms. Genetic Algorithms and Evolutionary Computation. Springer, New York (2002). https://doi.org/10.1007/978-1-4615-1539-5
26. Lehre, P.K., Nguyen, P.T.H.: On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In: Foundations of Genetic Algorithms, FOGA 2019, pp. 154–168. ACM (2019). https://doi.org/10.1145/3299904.3340316
27. Lengler, J., Sudholt, D., Witt, C.: The complex parameter landscape of the compact genetic algorithm. Algorithmica **83**(4), 1096–1137 (2020). https://doi.org/10.1007/s00453-020-00778-4
28. Mühlenbein, H., Paaß, G.: From recombination of genes to the estimation of distributions I. Binary parameters. In: Voigt, H.-M., Ebeling, W., Rechenberg, I., Schwefel, H.-P. (eds.) PPSN 1996. LNCS, vol. 1141, pp. 178–187. Springer, Heidelberg (1996). https://doi.org/10.1007/3-540-61723-X_982
29. Neumann, F., Witt, C.: Bioinspired Computation in Combinatorial Optimization - Algorithms and Their Computational Complexity. Springer (2010). https://doi.org/10.1007/978-3-642-16544-3
30. Ollivier, Y., Arnold, L., Auger, A., Hansen, N.: Information-geometric optimization algorithms: a unifying picture via invariance principles. J. Mach. Learn. Res. **18**, 1–65 (2017)
31. Pelikan, M., Hauschild, M.W., Lobo, F.G.: Estimation of distribution algorithms. In: Kacprzyk, J., Pedrycz, W. (eds.) Springer Handbook of Computational Intelligence, pp. 899–928. Springer, Heidelberg (2015). https://doi.org/10.1007/978-3-662-43505-2_45
32. Shapiro, J.L.: The sensitivity of PBIL to its learning rate, and how detailed balance can remove it. In: Foundations of Genetic Algorithms, FOGA 2002, pp. 115–132. Morgan Kaufmann (2002)
33. Shapiro, J.L.: Drift and scaling in estimation of distribution algorithms. Evol. Comput. **13**, 99–123 (2005). https://doi.org/10.1162/1063656053583414

34. Shapiro, J.L.: Diversity loss in general estimation of distribution algorithms. In: Runarsson, T.P., Beyer, H.-G., Burke, E., Merelo-Guervós, J.J., Whitley, L.D., Yao, X. (eds.) PPSN 2006. LNCS, vol. 4193, pp. 92–101. Springer, Heidelberg (2006). https://doi.org/10.1007/11844297_10
35. Stützle, T., Hoos, H.H.: MAX-MIN ant system. Futur. Gener. Comput. Syst. **16**, 889–914 (2000). https://doi.org/10.1016/S0167-739X(00)00043-1
36. Sudholt, D., Witt, C.: On the choice of the update strength in estimation-of-distribution algorithms and ant colony optimization. Algorithmica **81**, 1450–1489 (2019). https://doi.org/10.1007/s00453-018-0480-z
37. Witt, C.: Upper bounds on the running time of the univariate marginal distribution algorithm on onemax. Algorithmica **81**(2), 632–667 (2018). https://doi.org/10.1007/s00453-018-0463-0
38. Witt, C.: On crossing fitness valleys with majority-vote crossover and estimation-of-distribution algorithms. In: Foundations of Genetic Algorithms, FOGA 2021, pp. 2:1–2:15. ACM (2021). https://doi.org/10.1145/3450218.3477303
39. Zlochin, M., Birattari, M., Meuleau, N., Dorigo, M.: Model-based search for combinatorial optimization: a critical survey. Ann. Oper. Res. **131**, 373–395 (2004). https://doi.org/10.1023/B:ANOR.0000039526.52305.af