A Runtime Analysis for the Weighted Univariate Marginal Distribution Algorithm on LeadingOnes

Youssef Chaabouni[*]

Supervised by: Benjamin Doerr[†], Martin Krejca[†]

Abstract

The *univariate marginal distribution algorithm (UMDA)* is an established optimization heuristic for which there exist mathematical runtime guarantees. Recently, a more general class of algorithms has been introduced. We extend runtime results from the *UMDA* to a more general version of the algorithm, the *weighted UMDA* when optimizing the LeadingOnes benchmark in the generally desired regime with low genetic drift. We further simulate its behavior using two different weight distributions. We conjecture that the sum of the squared weights characterizes the runtime of the resulting algorithm.

1 Introduction

Heuristic optimization is a domain that focuses on optimization scenarios where no problem-specific algorithm exists. In these settings, algorithms such as evolutionary algorithms (*EA*s) [Sim13] are applied. An *EA* is an optimization algorithm that uses the principles of evolution found in nature, such as natural selection, to solve a given optimization problem. The candidate solutions to the problem are considered as individuals forming a population. As in natural selection, better individuals are selected. *EA*s are widely used for real-world problems, for example in the epistasis phenomenon [GZY$^+$19].

There are many different approaches that are classified as *EA*s. Estimation-of-distribution algorithms (*EDA*s) [PHL15] form an important

[*]École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

[†]Laboratoire d'Informatique (LIX), CNRS, École Polytechnique, Institut Polytechnique de Paris, Palaiseau, France

class of *EA*s that are different from the other approaches by the fact that the maintain a probabilistic model. To be more specific, an *EDA* aims to find the optimum of a given function on a set by building and iteratively evolving a probabilistic model on the search space. At each iteration, an *EDA* samples candidates from the search space based on the probabilistic model, then computes their function values and adjusts the model, giving more weight to the just created good samples. This way, we hope that the algorithm gets better at sampling optimal solutions.

The are many subclasses of *EA*s, and the fact that *EDA*s keep track of a probability distribution that evolves progressively in the iterations gives them advantages in certain areas. Many results have shown that *EDA*s, when used in the right way, can be very powerful compared to other *EA*s. For example, it is shown that they can have higher robustness to noise [FKKS16].

The good performance of *EDA*s, such as the *univariate marginal distribution algorithm (UMDA)* by Mühlenbein et al. [MP96], in real-world problems motivates a theoretical investigation. In the last few years, a lot of work has been done in the theoretical analysis of *EDA*s. Krejca and Witt give an overview of all the recent results [KW20]. One of the earliest papers is from Dang and Lehre [DL15] on the *univariate marginal distribution algorithm* optimizing the benchmarks OneMax and LeadingOnes by Rudolph [Rud97]. Since then, the runtime analysis for each of the benchmarks was improved independently. Improvements of the upper bound for OneMax are given by Lehre and Nguyen [LN17] and by Witt [Wit19]. On the other hand, the runtime bound for LeadingOnes is improved by Doerr and Krejca [DK21a].

In this work, we extend in Theorem 3 the runtime analysis of *UMDA* optimizing LeadingOnes to a more general version of the algorithm where the offspring are taken into consideration in a weighted manner when evolving the probabilistic model [DD22]. We then discuss different insights of this idea of a *weighted UMDA*. We believe our result improves the classical view of *UMDA* by bringing this new weights factor into consideration.

Then, we complement our theoretical results with experiments on *weighted UMDA* optimizing LeadingOnes using different weight distributions. We discuss the runtime for each of the distributions as a function of the selection ratio as this has been discussed on many occasions in the past [DK21a], [LN19]. Finally, we conjecture that the behavior of the *weighted UMDA* is characterized by the sum of the squared weights. That is, two instances of the *weighted UMDA* with different weight distributions but the same sum of the squared weights result in the same runtime.

2 Preliminaries

An *EDA* is an optimization algorithm that finds the optimum of a given *fitness function.*

Given a positive integer n that denotes the dimension of the problem, each bit-string (a string of 0s and 1s) of length n is called *individual*, and a multiset of individuals is called *population.* For example, when $n = 5$, 10010 is an individual and when $n = 2$, the population is $\{00,01,10,11\}$. We call *fitness function* any pseudo-Boolean function, that is any function $f : \{0,1\}^n \to \mathbb{R}$. Given an individual $x \in \{0,1\}^n$, we call $f(x)$ the *fitness* of x.

Now we define our algorithm of interest, the *weighted UMDA* (Algorithm 1) optimizing a pseudo-Boolean function. *UMDA* keeps a vector p of *frequencies* that it updates iteratively. Each frequency has a value of $p_i \in (0,1)$. At each iteration, the algorithm samples a population of λ individuals such that for each individual, the i^{th} bit is equal to 1 with probability p_i and 0 otherwise. It then computes the fitness of each individual and keeps only the μ individuals with the highest fitness values. Finally, it updates p by setting p_i to the weighted average of the i^{th} bit of those individuals it kept for all $i \in \{1..n\}$. It keeps going like that until a stopping criterion is met. In our case, we stop when a maximize is reached.

We note that a frequency close to 0 or 1 slows down the evolution of the algorithm as it decreases the diversity of the sample. In fact, if a frequency is close to 0, then it almost always samples a 0, meaning that it stays close 0, and the same happens for 1. The phenomenon that pulls the frequencies to 0 and 1 is called *genetic drift.* In order to make sure that no frequency gets stuck at 0 or 1, each time that a frequency is higher than $1 - \frac{1}{n}$ we set it to $1-\frac{1}{n}$, and each time that a frequency is smaller than $\frac{1}{n}$, we set it to $\frac{1}{n}$.

The fitness function of interest here is $\textsc{LeadingOnes} : \{0,1\}^n \to [0,n] \cap \mathbb{N}$ that, given an individual x, returns the length of the longest prefix of 1s of the bit-string x, that is:

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

We notice that LeadingOnes has a maximum value of n uniquely reached at the all-1s individual.

At any iteration t of the *weighted UMDA* optimizing LeadingOnes, a position i is called *critical* if all the frequencies preceding it are at $1 - \frac{1}{n}$ while $p_i^{(t)} < 1 - \frac{1}{n}$. Note that at most one position is critical at each iteration.

Algorithm 1 The weighted univariate marginal distribution algorithm (*weighted UMDA*) with parameters λ, μ and weights $\gamma_1 \geq \cdots \geq \gamma_\mu \geq \gamma_{\mu+1} = \cdots = \gamma_\lambda = 0$ such that $\sum_{i=1}^{\lambda} \gamma_i = 1$; optimizing a pseudo-Boolean function f.

```
$t \leftarrow 0$
$p^{(t)} \leftarrow (\frac{1}{2})_{i \in [n]}$
repeat ▷ iteration $t$
    for $i \in [\lambda]$ do  $x^{(i)} \leftarrow$ individual sampled via $p^{(t)}$;
    let $y^{(1)}, \ldots, y^{(\mu)}$ denote the $\mu$ individuals out of $x^{(1)}, \ldots, x^{(\lambda)}$ with the best
    fitness (breaking ties uniformly at random)
    for $i \in [n]$ do  $p_i^{(t+1)} \leftarrow \sum_{i=1}^{\mu} \gamma_i y_j^{(i)}$;
    restrict $p^{(t+1)}$ to the interval $[\frac{1}{n}, 1 - \frac{1}{n}]$
until termination criterion met;
```

3 Previous Work

The best known runtime upper bound for *UMDA* with parameters λ and μ can be summarized in the following theorem. This is the special case of the *weighted UMDA* where $\gamma_1 = \cdots = \gamma_\mu = \frac{1}{\mu}$ and $\gamma_{\mu+1} = \cdots = \gamma_\lambda = 0$.

Theorem 1. *(UMDA runtime upper bound [DK21a])*
Let $\delta \in (0, 1)$ be a constant, and let $\zeta = \frac{1-\delta}{4e}$. Consider UMDA optimizing LeadingOnes *with $\mu \geq 128 n \ln n$ and $\lambda \geq \frac{\mu}{\zeta}$. Further, let $d = \lfloor \log_4(\zeta \frac{\lambda}{\mu}) \rfloor$. Then UMDA samples the optimum after at most $\lambda(\lceil \frac{n}{d+1} \rceil + \lceil \frac{n}{n-1} e \ln n \rceil)$ fitness function evaluations with a probability of at least $1 - 5n^{-1}$.*

The result above is achieved by observing that in the low genetic drift regime, the frequencies are likely to stay close to 1/2. It follows that the total runtime in fitness evaluations is roughly equal to λ times the expected number of iterations we need to get all the critical frequencies to $1 - \frac{1}{n}$ consecutively.

In order to get a similar result for the *weighted UMDA*, we need to bound the negative effect of genetic drift for the general algorithm. The following result, recently proven by Doerr and Dufay in [DD22] gives an upper bound for this effect on the *weighted UMDA*. It states that given a pseudo-Boolean function that *weakly prefers* a 1 to a 0 at any position, meaning that $x_i = 1$ yields a fitness at least as good as $x_i = 0$ if all the other bits are maintained, then the frequencies of *weighted UMDA* stay close to $\frac{1}{2}$ for a long time.

Theorem 2. *(weighted UMDA genetic drift [DD22])*
Consider the UMDA with parameters λ, μ and $\gamma_1 \geq \cdots \geq \gamma_\mu \geq \gamma_{\mu+1} = \cdots = \gamma_\lambda = 0$. Assume it optimizes a function f that weakly prefers a 1 to 0 at

position i. For all $T \in \mathbb{N}$ and $\delta > 0$, we have:

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

In our context, we use the result above for LeadingOnes. Note that LeadingOnes weakly prefers a 1 to a 0.

4 Our Results

In the following, we present our generalized runtime analysis for *weighted UMDA* optimizing LeadingOnes, which gives the following theorem.

Theorem 3. *Let $\delta \in (0,1)$ be a constant, and let $\zeta = \frac{1-\delta}{4e}$. Consider the weighted UMDA optimizing LeadingOnes with $\sum_{i=1}^{\lambda}\gamma_i^2 \leq \frac{1}{128n\ln n}$ and $\lambda \geq \frac{\mu}{\zeta}$. Further, let $d = \lfloor \log_4(\zeta\frac{\lambda}{\mu})\rfloor$. Then the weighted UMDA samples the optimum after at most $\lambda(\lceil\frac{n}{d+1}\rceil + \lceil\frac{n}{n-1}e\ln n\rceil)$ fitness function evaluations with a probability of at least $1 - 5n^{-1}$.*

To show the result above, we follow the intuition in the proof of [DK21a, Theorem 5]. We first note that in a regime with low genetic drift, the frequencies must stay higher than a certain value, say $p_i > \frac{1}{4}$. Then we find that after each iteration, the critical frequency, all its preceding frequencies, and roughly the $\log\frac{\lambda}{\mu}$ following frequencies are set to $1 - \frac{1}{n}$ with high probability. Using a union bound on all failing cases, the result follows. The complete proof is in Appendix A.

The insight brought by Theorem 3 is that in the regime with low genetic drift where the inverse of the sum of the squared weights is at least quasilinear, the runtime of *weighted UMDA* in fitness evaluations is at most linear in the problem size divided by the logarithm of the selection rate. The fact that the condition above is only affected by the weight distribution through $\sum_{i=1}^{\lambda}\gamma_i^2$ motivates the next section.

5 Experiments

In this section, we run an experimental study on the behavior of the *weighted UMDA*. In particular, we study its runtime as a function of the ratio $\frac{\mu}{\lambda}$ as this ratio has been discussed on many occasions in the past [DK21a] and called the *selection pressure* by Lehre and Nguyen [LN19].

Looking at the *weighted UMDA*, one might wonder if the behavior or the algorithm depends on the distribution of the weights. For example, one might

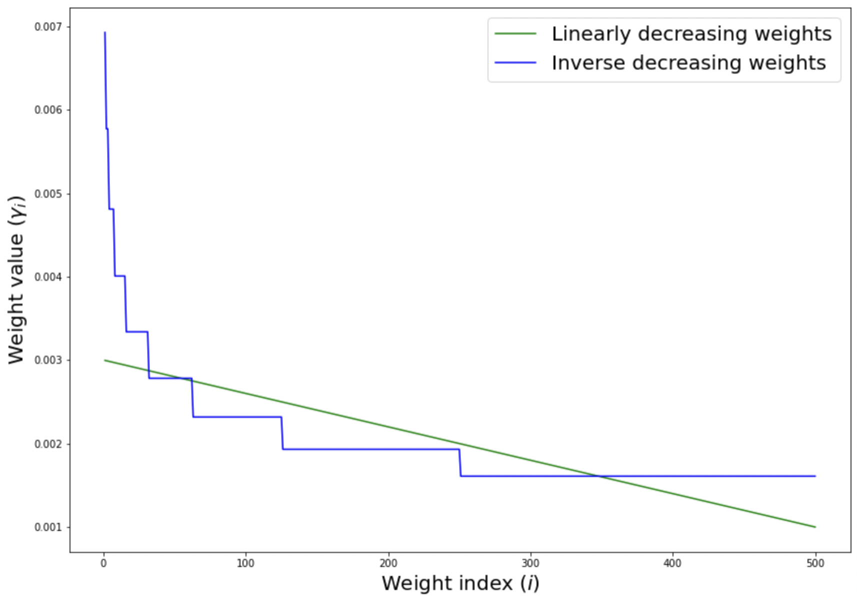

Figure 1: Weight distributions for $\mu = 500$

argue that the prior weights (corresponding to the individuals with higher fitness) should be affected by a higher value than later ones to make sure that this difference in fitness is taken into consideration.

In the following, we aim to compare the behaviors of the *weighted UMDA* using different weight distributions. For that, we start by introducing two models of weight distribution that we will use later in the simulations: the **linearly decreasing weights** and the **inverse decreasing weights** (see Figure 1). The exact definitions of these two weight distribution models can be found in Appendix B.

In order to study the effect of the weight distribution on the behavior of the algorithm, we cancel out the effect of genetic drift. In fact, when optimizing LeadingOnes, algorithms with higher genetic drift result in better runtimes. However, genetic drift is generally considered to be an undesired phenomenon as it can lead to drastic runtimes [DK21b]. Further details about how this is done and the experimental settings can be found in Appendix B.

Using the $n = 20$, $\lambda = 1000$ and values of μ ranging from 1 to 500, we obtain the results in Figure 2. By analyzing the curves, we deduce that for different values of $\sum_{i=1}^{\lambda} \gamma_i^2$, both the different weight distributions behave similarly. However, algorithms with a higher value of $\sum_{i=1}^{\lambda} \gamma_i^2$ have a lower runtime (which was expected since a high genetic drift results in a better runtime of *weighted UMDA* when optimizing LeadingOnes).

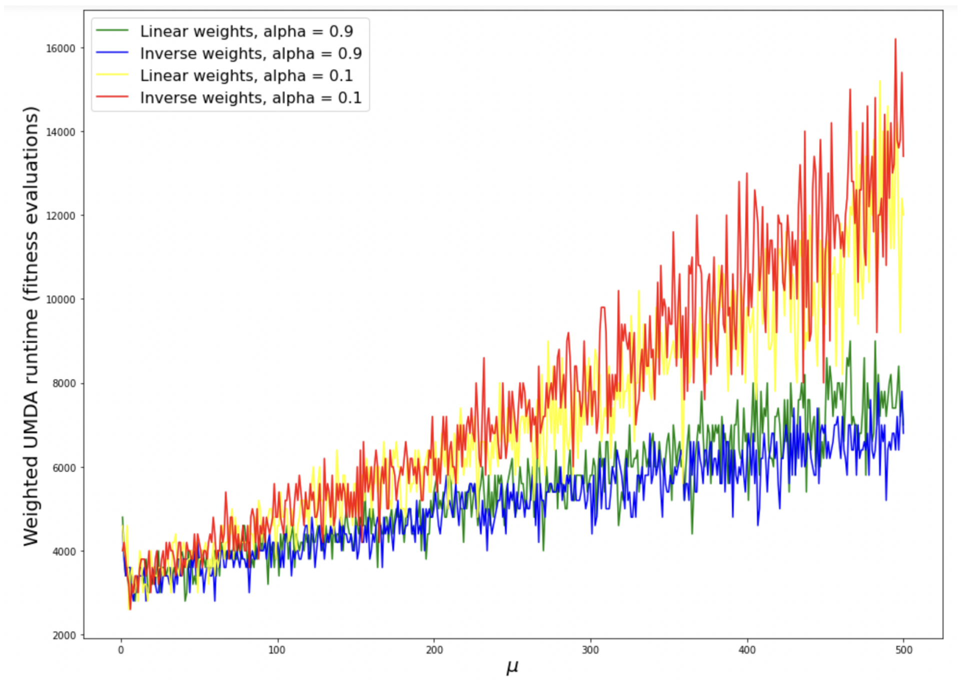

Figure 2: *Weighted UMDA* using different weight distributions

6 Conclusion & Future Work

On one hand, in Theorem 3 above, we generalize the runtime upper bound of the *univariate marginal distribution algorithm* to a more general weighted version of the algorithm which opens new doors for investigation. In the future, we suggest the exploration of the runtime for *Population Based Incremental Learning (PBIL)* which is a broader class of algorithms, and the even broader class of *EDA*s discussed by Dufay and Doerr in [DD22].

On the other hand, in view of the results obtained in the experiments above, we conjecture that the behavior of the *weighted UMDA* only depends on $\sum_{i=1}^{\lambda} \gamma_i^2$. We believe that the amount of genetic drift is deterministic of the behavior of the *weighted UMDA*. As we are currently working on the optimization of LeadingOnes, a higher genetic drift results in a better runtime, however, as discussed above, genetic drift is generally undesired. With that being said, we suggest making simulations using fitness functions where a high genetic drift can lead to a drastically high runtime, for example, the DeceptiveLeadingBlocks benchmark introduced by Lehre and Nguyen [LN19].

References

[DD22] Benjamin Doerr and Marc Dufay. General univariate estimation-of-distribution algorithms. In Günter Rudolph, Anna V. Kononova, Hernán E. Aguirre, Pascal Kerschke, Gabriela Ochoa, and Tea Tusar, editors, *Parallel Problem Solving From Nature, PPSN 2022*, pages 470–484. Springer, 2022.

[DK21a] Benjamin Doerr and Martin S. Krejca. A simplified run time analysis of the univariate marginal distribution algorithm on LeadingOnes. *Theoretical Computer Science*, 851:121–128, 2021.

[DK21b] Benjamin Doerr and Martin S. Krejca. The univariate marginal distribution algorithm copes well with deception and epistasis. *Evolutionary Computation*, 29:543–563, 2021.

[DL15] Duc-Cuong Dang and Per Kristian Lehre. Simplified runtime analysis of estimation of distribution algorithms. In *Genetic and Evolutionary Computation Conference, GECCO 2015*, pages 513–518. ACM, 2015.

[FKKS16] Tobias Friedrich, Timo Kötzing, Martin S. Krejca, and Andrew M. Sutton. Robustness of ant colony optimization to noise. *Evolutionary Computation*, 24:237–254, 2016.

[GZY$^+$19] Yang Guo, Zhiman Zhong, Chen Yang, Jiangfeng Hu, Yaling Jiang, Zizhen Liang, Hui Gao, and Jianxiao Liu. Epi-GTBN: an approach of epistasis mining based on genetic Tabu algorithm and Bayesian network. *BMC Bioinformatics*, 20, 444, 2019.

[KW20] Martin S. Krejca and Carsten Witt. Theory of estimation-of-distribution algorithms. In Benjamin Doerr and Frank Neumann, editors, *Theory of Evolutionary Computation: Recent Developments in Discrete Optimization*, pages 405–442. Springer, 2020. Also available at https://arxiv.org/abs/1806.05392.

[LN17] Per Kristian Lehre and Phan Trung Hai Nguyen. Improved runtime bounds for the univariate marginal distribution algorithm via anti-concentration. In *Genetic and Evolutionary Computation Conference, GECCO 2017*, pages 1383–1390. ACM, 2017.

[LN19] Per Kristian Lehre and Phan Trung Hai Nguyen. On the limitations of the univariate marginal distribution algorithm to deception and where bivariate EDAs might help. In *Foundations of Genetic Algorithms, FOGA 2019*, pages 154–168. ACM, 2019.

[MP96] Heinz Mühlenbein and Gerhard Paass. From recombination of genes to the estimation of distributions I. Binary parameters. In *Parallel Problem Solving from Nature, PPSN 1996*, pages 178–187. Springer, 1996.

[PHL15] Martin Pelikan, Mark Hauschild, and Fernando G. Lobo. Estimation of distribution algorithms. In Janusz Kacprzyk and Witold Pedrycz, editors, *Springer Handbook of Computational Intelligence*, pages 899–928. Springer, 2015.

[Rud97] Günter Rudolph. *Convergence Properties of Evolutionary Algorithms.* Verlag Dr. Kovăc, 1997.

[Sim13] Dan Simon. *Evolutionary Optimization Algorithms.* John Wiley & Sons, 2013.

[Wit19] Carsten Witt. Upper bounds on the running time of the univariate marginal distribution algorithm on OneMax. *Algorithmica*, 81:632–667, 2019.

A Omitted proofs

Proof of Theorem 3:

Lemma 4. *Consider the weighted UMDA. Assume that it optimizes a function that weakly prefers a 1 at all positions with* $\sum_{i=1}^{\lambda} \gamma_i^2 \leq \frac{1}{128n \ln n}$*. Then, with a probability of at least* $1 - 2n^{-1}$*, each frequency stays at a value of at least* $\frac{1}{4}$ *for the first* $2n$ *iterations.*

Proof. Using Theorem 2 with $\delta = \frac{1}{4}$ and $T \leq 2n$ we obtain:

$$\begin{aligned} P[\forall t \in [0..T], |p_i^{(t)} - 1/2| < \delta] &\geq 1 - 2\exp\left(\frac{-\delta^2}{2T\sum_{i=1}^{\lambda} \gamma_i^2}\right) &(1)\\ &\geq 1 - 2e^{-2\ln n} &(2)\\ &= 1 - 2n^{-2} &(3)\end{aligned}$$

The inequality above says that we have a failure probability of $2n^{-2}$. Taking the union bound over this probability for each frequency gives an overall failure probability of at most $n \times 2n^{-2} = 2n^{-1}$. The result follows. □

Lemma 5. *Let* $\delta \in (0,1)$ *be a constant, and let* $\zeta = \frac{1-\delta}{4e}$*. Consider the weighted UMDA optimizing* LeadingOnes *with* $\mu \geq 4\frac{1-\delta}{\delta^2}\ln n$ *and* $\lambda \geq \frac{\mu}{\lambda}$*. Furthermore, consider an iteration* $t \in \mathbb{N}$ *such that position* $i \in [n]$ *is critical and that, for all positions* $j \geq i$*, we have* $p_j^{(t)} \geq \frac{1}{4}$*. Let* $d = \lfloor \log_4(\zeta\frac{\lambda}{\mu}) \rfloor$*. Then, with a probability of at least* $1 - n^{-2}$*, for all positions* $j \in [\min\{n, i + d\}]$*, we have* $p_j^{(t+1)} = 1 - \frac{1}{n}$*.*

Proof. We use a similar argument to the proof of [DK21a, Lemma 2], replace the bound of μ by the one for $\sum_{i=1}^{\lambda} \gamma_i^2$. □

Using the two lemmas above, we prove our main result.

Proof of Theorem 3. We use a similar argument to the proof of [DK21a, Theorem 5]. We know that LeadingOnes weakly prefers a 1 to a 0. By Lemma 4 , since $\sum_{i=1}^{\lambda} \gamma_i^2 \leq \frac{1}{128n \ln n}$ then all frequencies stay above $\frac{1}{4}$ for $2n$ iterations with probability of at least $1 - 2n^{-2}$.
Using Lemma 5, we note that at least $d + 1$ additional frequencies are set to $1 - \frac{1}{n}$ with a probability of at least $1 - n^{-2}$. Using a union bound on the first $2n$ iterations, this implies that after $\lceil \frac{n}{d+1} \rceil$ all of the frequencies are at $1 - \frac{1}{n}$ with a probability of at least $1 - 2n^{-1}$.
Once all the frequencies are at $1 - \frac{1}{n}$, the probability of sampling the optimum at each iteration is $\left(1 - \frac{1}{n}\right)^n \geq \left(1 - \frac{1}{n}\right)\frac{1}{e}$. After $\lceil \frac{n}{n-1} e \ln n \rceil$ iterations, the probability of having sampled the optimum is $1 - \left(1 - \frac{n-1}{n}\frac{1}{e}\right)^{\lceil \frac{n}{n-1} e \ln n \rceil} \geq 1 - n^{-1}$.

To sum up, we use union bound on all the failure events seen above to prove that the optimum is sampled after $\lceil \frac{n}{d+1} \rceil + \lceil \frac{n}{n-1} e \ln n \rceil$ with a probability of a least $1 - 5n^{-1}$. Finally, since each iteration is worth λ fitness evaluations, we deduce that the *weighted UMDA* needs at most $\lambda \Big(\lceil \frac{n}{d+1} \rceil + \lceil \frac{n}{n-1} e \ln n \rceil \Big)$ with a probability of a least $1 - 5n^{-1}$. □

B Experimental Settings

We start by introducing the two models of weight distributions used in the simulations (see Figure 1).

- **Linearly decreasing weights:**
  The weights are decreasing linearly in the index, that is:

  $$\gamma_i = \begin{cases} -ai + b & \text{if } 1 \leq i \leq \mu \\ 0 & \text{else} \end{cases}$$

  for some $b \in \left[\frac{1}{\mu}, \frac{2}{\mu}\right]$ and $a > 0$ chosen such that $\sum_{i=1}^{\lambda} \gamma_i = 1$.

- **Inverse decreasing weights:**
  The weights are taken such that:

  $$\forall i \in \left(\frac{\mu}{2^{n+1}}, \frac{\mu}{2^n} \right] \text{ we have } \gamma_i = \alpha \beta^n$$

  for some $\beta > 1$ and $\alpha > 0$ chosen such that $\sum_{i=1}^{\lambda} \gamma_i = 1$.

As mentioned above, we cancel out the effect of genetic drift in order to study the effect of the weight distribution on the behavior of *weighted UMDA*. In fact, we have seen in Theorem 2 that $\sum_{i=1}^{\lambda} \gamma_i^2$ provides a bound for genetic drift. Based on that, we use the sum of squared weights to characterize the amount of genetic drift. For each of the two weight distributions introduced above, we run simulations and plot the runtime of *weighted UMDA* optimizing LeadingOnes as a function of μ, using a fixed λ. As discussed above, we want to make sure that the amount of genetic drift is the same when comparing the algorithms of different weight distributions. We do that by generating the **linearly decreasing weights** and then finding the parameters of the **inverse decreasing weights** that would lead to the same value of $\sum_{i=1}^{\lambda} \gamma_i^2$ in both weights distributions. To be more specific, for a given value of $\alpha \in (0, 1)$, we generate the linearly decreasing weights such that $\gamma_1 = \frac{(1+\alpha)}{\mu}$ and $\gamma_\mu = \frac{(1-\alpha)}{\mu}$, then find the parameters of the inverse decreasing weights such that the sums of the squared weights are equal for the two sequences of weights.

The plots in Figure 2 are generated using $\alpha = 0.1$ and $\alpha = 0.9$. For each $\mu \in \{1 \dots 500\}$, we take the average of the runtimes of 10 runs of *weighted UMDA* optimizing LeadingOnes with parameters $n = 20$, $\lambda = 1000$, μ and $\gamma_1 \dots \gamma_\lambda$ of the corresponding weight distribution.