Self-adjusting mutation rates with provably optimal success rules

Benjamin Doerr, Carola Doerr, Johannes Lengler

Self-Adjusting Mutation Rates with Provably Optimal Success Rules

Benjamin Doerr
École Polytechnique, CNRS, LIX
Palaiseau, France

Carola Doerr
Sorbonne Université, CNRS, LIP6
Paris, France

Johannes Lengler
ETH
Zürich, Switzerland

ABSTRACT

The one-fifth success rule is one of the best-known and most widely accepted techniques to control the parameters of evolutionary algorithms. While it is often applied in the literal sense, a common interpretation sees the one-fifth success rule as a family of success-based updated rules that are determined by an update strength F and a success rate s. We analyze in this work how the performance of the (1+1) Evolutionary Algorithm (EA) on LeadingOnes depends on these two hyper-parameters. Our main result shows that the best performance is obtained for small update strengths $F = 1+o(1)$ and success rate $1/e$. We also prove that the runtime obtained by this parameter setting is asymptotically optimal among all dynamic choices of the mutation rate for the (1+1) EA, up to lower order error terms. We show similar results for the resampling variant of the (1+1) EA, which enforces to flip at least one bit per iteration.

CCS CONCEPTS

• **Theory of computation → Random search heuristics**;

1 INTRODUCTION

One of the key challenges in applying evolutionary algorithms (EAs) in practice lies in suitable choices of the population sizes, the mutation rates, crossover probabilities, selective pressure, and possibly other parameters that determine the exact structure of the heuristic. What complicates the situation is that the optimal values of these parameters may change during the optimization process, so that an ideal parameter setting requires to find not only good initial values, but also suitable update rules that adjust the parameters during the run. *Parameter control* is the umbrella term under which such non-static parameter settings are studied. Parameter control is indispensable in continuous optimization, where the step size needs to be adjusted in order to obtain good convergence to the optima, and is standard since the early seventies. In discrete optimization, however, parameter control has received much less attention, as commented in the recent surveys [1, 21]. This situation has changed substantially in the last decade, both thanks to considerable advances in reinforcement learning, which could be successfully leveraged to control algorithmic parameters [6, 18, 20], but also thanks to a number of theoretical results rigorously quantifying the advantages of dynamic parameter settings over static ones, cf. [10] for a summary of known results.

One of the best known parameter update rules is the *one-fifth success rule*, which was independently designed in [7, 25, 26]. The one-fifth success rule states that it is desirable to maintain a success rate, measured by the frequency of offspring having a better fitness than the current-best individual, of around 20%. Theoretical justification for this rule was given by Rechenberg, who showed that such a success rate is optimal for controlling the step size of the (1+1) Evolution Strategy (ES) optimizing the sphere function [25]. Based on this finding several parameter update rules were designed that decrease the step size when the observed success rate is smaller than this target rate, and which increase it for success rates larger than 20%.

An interpretation of the one-fifth success rule which is suitable also for parameter control in discrete domains was provided in [22]. Kern *et al.* propose to decrease the step size σ to σ/F after each successful iteration, and to increase it to $\sigma F^{1/4}$ otherwise. They propose to consider an iteration successful if the offspring y created in this iteration is at least as good as its parent x, i.e., if $f(y) \le f(x)$ in the context of minimizing the function f. With this rule, the step size remains constant when one out of five iterations is successful, since in this case after the fifth iteration σ has been replaced by $\sigma \cdot (F^{1/4})^4/F$. This version of the one-fifth success rule, typically using constant update strengths $F > 1$, was shown to work efficiently, e.g., in [2]. In [9] it was proven to yield asymptotically optimal linear expected optimization time when applied to the $(1 + (\lambda, \lambda))$ GA optimizing OneMax. No static parameter choice can achieve this efficiency, since all static variants of the $(1+(\lambda, \lambda))$ Genetic Algorithm (GA) require super-linear runtimes [9].

Other success-based multiplicative update rules had previously been studied in the theory of evolutionary algorithms (EAs). For example, Lässig and Sudholt [23] showed that for four classic benchmark problems the expected number of generations needed to find an optimal solution is significantly reduced when multiplying the offspring population size λ by two after every unsuccessful iteration of the $(1 + \lambda)$ EA and reducing λ to $\lambda/2$ otherwise. Similar rules which also take into account the number of improved offspring were empirically shown to be efficient in [19]. Recently, Doerr and Wagner [17] showed that success-based multiplicative updates are very efficient for controlling the mutation rate of the $(1 + 1)$ EA$_{>0}$, the $(1 + 1)$ EA variant proposed in [4], which enforces to flip at least one bit per each iteration. More precisely, they analyze the average optimization times of the $(1 + 1)$ EA(A, b) algorithm which increases the mutation rate p by a factor of $A > 1$ when the offspring y satisfies $f(y) \ge f(x)$ (i.e., when it replaces its parent x) and which decreases p to bp, $0 < b < 1$ otherwise. Their experimental results show that this algorithm for broad ranges of A and b has a good performance on OneMax and LeadingOnes.

Our Results. In this work, we complement the empirical study [17] and rigorously prove that for suitably chosen hyper-parameters A and b the $(1+1)$ EA using this multiplicative update scheme has an asymptotically optimal expected runtime on the LeadingOnes function $\text{Lo} : \{0,1\}^n \to [0..n] = \{0\} \cup \mathbb{N}_{\leq n}, x \mapsto \max\{i \in [0..n] \mid \forall j \leq i : x_j = 1\}$, where in this work we refer to a runtime as "asymptotically optimal" when it is optimal up to lower order terms among all dynamic choices of the mutation rate. For the $(1+1)$ EA$_{>0}$ we also rigorously prove a bound on the expected optimization time on LeadingOnes, which we show by numerical evaluations to coincide almost perfectly with the performance achieved by the best possible $(1+1)$ EA$_{>0}$ with optimally controlled mutation rates.

Following the suggestion made in [22], and adapting to the common notation, we formulate our theoretical results using the parametrization $A = F^s$ and $b = 1/F$, where F denotes again the update strength and s the *success ratio*. As seen above, a success *ratio* of 4 corresponds to a one-fifth success *rule*.

We prove that for the $(1+1)$ EA the optimal success ratio is $e-1$ (i.e., a $1/e$ success rule). More precisely, we show that the expected runtime of the self-adjusting $(1+1)$ EA with constant success ratio $s > 0$ and small update strength $F = 1 + o(1)$ on LeadingOnes is at most $\frac{s+1}{4\ln(s+1)}n^2 + o(n^2)$. The expected runtime with $s = e-1$ is asymptotically optimal among all $(1+1)$ EA variants that differ only in the choice of the mutation rates. A key ingredient in this proof is a lemma proving that the mutation rate used by the $(1+1)$ EA with self-adjusting mutation rates is, at all times during the optimization process, very close to the *target mutation rate* $\rho^*(\text{Lo}(x), s) \approx \ln(s+1)/\text{Lo}(x)$, which we define as the unique mutation rate that leads to success probability $1/(s+1)$.

We also extend our findings to the $(1+1)$ EA$_{>0}$ considered in [17]. This resampling $(1+1)$ EA variant is technically more challenging to analyze, since the probabilities of the conditional standard bit mutation operator (which enforces to flip one bit) are more complex to handle, but also because the concept of target mutation rates ceases to exist for fitness levels $\ell \geq \frac{s}{s+1}n$, since it is impossible to achieve success rates of $1/(s+1)$ or higher for such values of ℓ without accepting duplicates as offspring. In this regime the mutation rate approaches zero, and the $(1+1)$ EA$_{>0}$ resembles Randomized Local Search (RLS), which flips in each iteration exactly one bit. This behavior is desirable since the optimal number of bits to flip in solutions x with $\text{Lo}(x) \geq n/2$ is indeed equal to one. In contrast to the unconditional $(1+1)$ EA, our bound for the expected runtime of the self-adjusting $(1+1)$ EA$_{>0}$ does not have a straightforward closed-form expression. A numerical evaluation for dimensions up to $n = 10\,000$ shows that the best runtime is achieved for success ratio $s \approx 1.285$. With this choice (and using again $F = 1 + o(1)$), the performance of the self-adjusting $(1+1)$ EA$_{>0}$ is almost indistinguishable from $(1+1)$ EA$_{>0,\text{opt}}$, the best possible $(1+1)$ EA$_{>0}$ variant using in each iteration the optimal mutation rate. Both algorithms achieve an expected runtime which is around $0.404n^2$.

For both algorithms, the self-adjusting $(1+1)$ EA and the $(1+1)$ EA$_{>0}$, we do not only bound the expected runtime but prove also stochastic domination bounds, which provide much more information about the runtime [8]. We only show upper bounds in this work, but we strongly believe that our bounds are tight, since for the $(1+1)$ EA we obtain asymptotically optimal runtime, and for the self-adjusting $(1+1)$ EA$_{>0}$ the numerical bounds are almost indistinguishable from those of $(1+1)$ EA$_{>0,\text{opt}}$.

Related Work. In [15, 24] variants of RLS flipping a dynamic number of bits were analyzed. These schemes, which take inspiration from the literature on hyper-heuristics, differ from our dynamic setting in particular in the fact that they consider only a constant-size set of possible parameter values. The mentioned analysis of the $(1 + (\lambda, \lambda))$ GA using the one-fifth success rule presented in [9] deviates from ours in that it only considers the order of magnitude, but not the leading constants of the runtime.

Availability of Full Proofs. Full proofs and numerical data for Fig. 3 are available in [12].

2 THE SELF-ADJUSTING (1+1) EA

We study the optimization time of the $(1+1)$ EA with self-adjusting mutation rates, Algorithm 1. This algorithm starts the optimization process with an initial mutation rate $\rho = \rho_0$ and a random initial solution $x \in \{0,1\}^n$. In every iteration one new solution candidate $y \in \{0,1\}^n$ is created from the current-best solution through standard bit mutation with mutation rate ρ, i.e., y is created from x by flipping each bit, independently of all other decisions, with probability ρ. If y is at least as good as its parent x, i.e., if $f(y) \geq f(x)$, x is replaced by its offspring y and the mutation rate ρ is increased to $\min\{F^s\rho, \rho_{\max}\}$, where $F > 1$ and $s > 0$ are two constants that remain fixed during the execution of the algorithm and $0 < \rho_{\max} \leq 1$ is an upper bound for the range of admissible mutation rates. If, on the other hand, y is strictly worse than its parent x, y is discarded and the mutation rate decreased to $\max\{\rho/F, \rho_{\min}\}$, where $0 < \rho_{\min}$ is the smallest admissible mutation rate. The algorithm continues until some stopping criterion is met. Since in our theoretical analysis we know the optimal function value $f_{\max}$, we use as stopping criterion that $f(x) = f_{\max}$. We call the number of function evaluations until an optimum is found the *runtime* or *optimization time* of the algorithm.

Standard Bit Mutation. Since we will also consider the $(1+1)$ EA$_{>0}$, which requires that each offspring y differs from its parent x in at least one bit, we use in lines 3 and 4 the equivalent description of standard bit mutation, in which we first sample the number k of bits to flip and then apply the mutation operator mut_k, which flips exactly k uniformly chosen bits in x.

Success Ratio vs. Success Rule. We recall from the introduction that we call F the *update strength* of the self-adjustment and s the *success ratio*. The success ratio $s = 4$ is particularly common in evolutionary computation [2, 9, 22], and is referred to as the *one-fifth success rule*: if one out of five iterations is successful, the parameter ρ stays constant. This rule was developed in [22] as a discrete analog of the one-fifth success rule known from evolution strategies [25]. Note that a success *ratio* of s corresponds to a $1/(s+1)$-th success *rule*. We choose to work with success ratios for notational convenience.

Hyper-Parameters. Altogether, the self-adjusting $(1+1)$ EA has five *hyper-parameters*: the update strength F, the success rate s,

Algorithm 1: The self-adjusting $(1+1)$ EA with update strength F, success ratio s, initial mutation rate ρ_0, minimal mutation rate $\rho_{\min}$, and maximal mutation rate $\rho_{\max}$. The formulation assumes maximization of the function $f : \{0,1\}^n \to \mathbb{R}$ as objective.

```
1  Initialization: Sample $x \in \{0,1\}^n$ uniformly at random
     and compute $f(x)$;
2  Set $\rho = \rho_0$;
3  Optimization: for $t = 1, 2, 3, \ldots$ do
4      Sample $k$ from $\mathrm{Bin}(n, \rho)$;
5      $y \leftarrow \mathrm{mut}_k(x)$;
6      evaluate $f(y)$;
7      if $f(y) \geq f(x)$ then
8          $x \leftarrow y$ and $\rho \leftarrow \min\{F^s\rho, \rho_{\max}\}$
9      else
10         $\rho \leftarrow \max\{\rho/F, \rho_{\min}\}$
```

the initial mutation rate ρ_0, and the minimal and maximal mutation rates $\rho_{\min}$ and $\rho_{\max}$, respectively. It is not difficult to verify that for update strengths $F = 1 + \varepsilon$, $\varepsilon = \Omega(1)$, the mutation rate deviates, in at least a constant fraction of all iterations, from the optimal one by at least a constant factor, which results in a constant factor overhead in the runtime. We therefore consider $F = 1 + o(1)$ only. Apart from this, we only require that $\rho_{\min} = o(n) \cap \omega(n^{-c})$ for an arbitrary constant c. In a practical application, $\rho_{\min} = n^{-2}$ appears to be a good choice. In our analysis, there is no reason to cap the values of ρ by using a $\rho_{\max}$ value strictly less than 1, so we always use $\rho_{\max} = 1$. A mutation rate of 1, clearly, does not make much sense in almost all cases, but in our analysis the self-adjustment automatically avoids such large ρ-values.

We easily see that Algorithm 1 generalizes the $(1+1)$ EA with static mutation rate ρ, which we obtain by setting $F = 1$ and $\rho_0 = \rho$.

Improvement vs. Success Probability. We study in this work the performance of the self-adjusting $(1+1)$ EA on the LeadingOnes function $\textsc{Lo} : \{0,1\}^n \to \mathbb{R}, x \mapsto \max\{j \in [0..n] \mid \forall i \leq j : x_j = 1\}$, which counts the number of initial ones in a bit string. We build our analysis on results presented in [3, 8], which reduce the study of the overall runtime to analyzing the time spent on each fitness level. More precisely, for a random solution $x \in \{0,1\}^n$ with $f(x) =: \ell$ we study the time T_ℓ that it takes the self-adjusting $(1+1)$ EA to reach for the first time a solution y of fitness $f(y) > \ell$. We call the probability to create such a y the *improvement probability* $p_{\mathrm{imp}}(\rho, \ell)$ of mutation rate ρ on level ℓ. For fixed mutation rate ρ, this improvement probability is easily seen to equal $(1-\rho)^\ell \rho$, since the first ℓ bits should not flip, the $(\ell+1)$-st should, and it does not matter what happens in the *tail* of the string.

Another important probability is the *success probability* $p_{\mathrm{suc}}(\rho, \ell) := (1-\rho)^\ell$ of creating an offspring y that is *at least as good* as x, since this is the probability of increasing the mutation rate from ρ to $\min\{F^s\rho, \rho_{\max}\}$.

We note that several other works studying self-adjusting parameter choices assume that the adjustment rule distinguishes whether or not a strict improvement has been found. In the analysis of the self-adjusting $(1 + (\lambda, \lambda))$ GA in [9], for example, it is assumed that $\lambda \leftarrow \lambda/F$ if and only if $f(y) > f(x)$, while $\lambda \leftarrow \lambda F^{1/4}$ otherwise (whereas, as recommended in [11], it is suggested to update x whenever $f(y) \geq f(x)$, so that a distinction between the parameter update and the selection step has to be made). Analyzing the effects of this choice goes beyond the scope of this present work, but it is certainly desirable to develop general guidelines which update rule to prefer for which type of problems.

2.1 Main Result

Theorem 2.1 summarizes the main result of this section. Before providing the formal statement, we introduce a quantity that will play an important role in all our computation, the *target mutation rate* $\rho^*(\ell, s)$. We consider as target mutation rate the value of p which leads to the success probability that is given by the success rule. That is, for each fitness level $1 \leq \ell \leq n-1$ and each success ratio $s > 0$ the target mutation rate $\rho^*(\ell, s)$ is the unique value $p \in (0,1)$ that satisfies $p_{\mathrm{suc}}(p) = (1-p)^\ell = 1/(s+1)$. For $\ell = 0$ we set $\rho^*(\ell, s) := 1$. A key argument in the following proofs will be that the mutation rate drifts towards this target rate.

Following the discussion in [8] we do not only analyze in Theorem 2.1 the expected runtime, but rather show a *stochastic domination* result. To formulate our results, we introduce the shorthand $X \preceq Y$ to express that the random variable X is stochastically dominated by the random variable Y, that is, that $\mathbb{P}[X \geq \lambda] \leq \mathbb{P}[Y \geq \lambda]$ for all $\lambda \in \mathbb{R}$. We also recall that a random variable X has a geometric distribution with success rate p, written as $X \sim \mathrm{Geom}(p)$, when $\mathbb{P}[X = k] = (1-p)^{k-1}p$ for all $k = 1, 2, \ldots$.

Theorem 2.1. *Let $c > 1$ be a constant. Consider a run of the self-adjusting $(1+1)$ EA with $F = 1 + \varepsilon$, $\varepsilon \in \omega(\log n/n) \cap o(1)$, $s > 0$, $\rho_{\min} \in o(n^{-1}) \cap \Omega(n^{-c})$, $\rho_{\max} = 1$, and arbitrary initial rate $\rho_0 \in [\rho_{\min}, \rho_{\max}]$ on the n-dimensional LeadingOnes function. Then the number T of iterations until the optimum is found is stochastically dominated by*

$$o(n^2) + \sum_{\ell=0}^{n-1} X_\ell \,\mathrm{Geom}(\min\{\omega(\tfrac{1}{n}), (1-o(1))(1-\rho^*(\ell,s))^\ell \rho^*(\ell,s)\}),$$

where the X_ℓ are uniformly distributed binary random variables and all X_ℓ and geometric random variables are mutually independent. Further, all asymptotic notation solely is with respect to n and can be chosen uniformly for all ℓ. In particular,

$$\begin{aligned} E[T] &\leq (1+o(1))\frac{1}{2}\sum_{\ell=0}^{n-1}\Big((1-\rho^*(\ell,s))^\ell \rho^*(\ell,s)\Big)^{-1} \\ &= (1+o(1))\frac{s+1}{4\ln(s+1)}n^2. \qquad (1) \end{aligned}$$

2.2 Numerical Evaluation

Fig. 1 displays the by n^2 normalized expected optimization time $\frac{s+1}{4\ln(s+1)}$ for success ratios $0 \leq s \leq 10$. Minimizing this expression for s shows that a success ratio of $s = e - 1$ is optimal. With this setting, the self-adjusting $(1+1)$ EA yields an expected optimization time of $(1 \pm o(1))en^2/4$, which was shown in [3] to be optimal across all possible adaptive $(1+1)$ EA variants. In fact, with this success ratio, it holds that $\rho^*(\ell, s) \approx 1/(\ell+1)$, which is the mutation rate

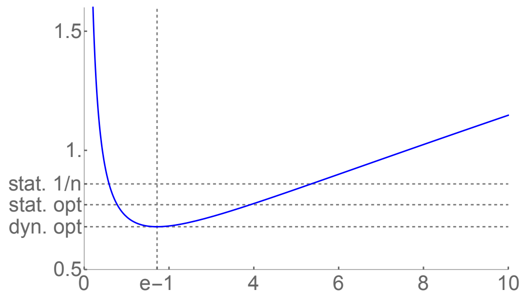

Figure 1: Normalized (by n^2) expected optimization times of the self-adjusting $(1+1)$ EA for different success ratios s, and assuming $F = 1 + o(1)$.

that was shown in [3] to be the optimal rate for random solutions with $\mathrm{Lo}(x) = \ell$.

Using Inequality (1) we also observe that for all success ratios $s \in [0.78, 3.92]$ the expected optimization time of the self-adjusting $(1+1)$ EA is better than the $0.77201n^2$ one of the best static $(1+1)$ EA computed in [3], which uses mutation rate $p^* = 1.5936.../n$. Note also that the one-fifth success rule (i.e., $s = 4$) performs slightly worse; its expected optimization time is around $0.7767n^2$. Note, however, that we will see in Section 4 (cf. also Fig. 3), that its fixed-target performance is nevertheless better than the static one with p^* for a large range of sub-optimal target values.

Finally, we note that for success ratios $s \in [0.59, 5.35]$ the expected optimization time of the self-adjusting $(1+1)$ EA is better than the $0.85914..,n^2$ one of the static $(1+1)$ EA with default mutation rate $p = 1/n$.

2.3 Occupation Probabilities

In our proofs, we will need the following result showing that a random process with negative additive drift in the non-negative numbers cannot often reach states that are mildly far in the positive numbers. Results of a similar flavor have previously been obtained in [16], but we do not see how to derive our result easily from this work.

Lemma 2.2. *Let D be a discrete random variable satisfying $|D| \leq s$ and $E[D] = -\delta$ for some $\delta \leq \sqrt{2}s$. Let X_t be a random process on $\mathbb{R}$ such that*

- *(X_t) starts on a fixed value $x_0 \leq s$, that is, we have $\mathbb{P}[X_0 = x_0] = 1$,*
- *for all $t \geq 1$ and for all $r_1, \ldots, r_t \in \mathbb{R}$ with $\mathbb{P}[\forall i \in [1..t] : X_i = r_i] > 0$ we have*
  - *if $r_t \geq 0$, then conditioned on $X_1 = r_1, \ldots, X_t = r_t$ the conditional distribution of X_{t+1} is dominated by $r_t + D$,*
  - *if $r_t < 0$, then $X_{t+1} - X_t$ has a discrete distribution with absolute value at most s.*

Then for all $t \geq 1$ and $U \geq s$, we have

$$\mathbb{P}[X_t \geq U] \leq \exp\left(-\frac{\delta(U-s)}{2s^2}\right)\left(\frac{U-s}{\delta} + 1 + \frac{4s^2}{\delta^2}\right).$$

In particular, for $U = 6s^2 \ln(1/\delta)/\delta + s$, we have $\mathbb{P}[X_t \geq U] \leq 6\delta s^2 \ln(1/\delta) + \delta^3 + 4\delta s^2$, an expression tending to zero for $\delta \to 0$.

2.4 Proof Overview

The main proof idea consists in showing that in a run of the self-adjusting $(1+1)$ EA we sufficiently often have a mutation rate that is close to the target mutation rate (the unique rate which gives a success rate of $1/(s+1)$). We obtain this information from exhibiting that the self-adjustment leads to a drift of the mutation rate towards the target rate. This drift is strong when the rate is far from the target, so we can use a multiplicative drift argument to show that the rate quickly comes close to the target rate (Lemma 2.5). Once close, we use our occupation probability lemma (Lemma 2.2) based on additive drift to argue that the rate often is at least mildly close to the target (Lemma 2.6). We need a careful definition of the lower order expressions “often”, “close”, and “mildly close” to make this work.

From the knowledge that the rate is often at least mildly close to the target rate, we would like to derive that the optimization process is similar to using the target rate in each iteration. This is again not trivial and a main obstacle is that the rate is not chosen independently in each iteration. Consequently, we cannot argue that each iteration on one fitness level has the same, independent probability for finding an improvement (which would give that the waiting time on the level follows a geometric distribution). We overcome this difficulty by splitting the time spent on one fitness level in short independent phase each consisting of bringing the rate into the desired region and then exploiting that the rate will stay there most of the time (Lemma 2.8). This approach is feasible because of our relatively good bounds for the time needed to reach the desired rate range. The final argument is that the runtime of the self-adjusting $(1+1)$ EA on LeadingOnes is half the sum of the times needed to leave each fitness level. Such a statement has been previously observed for static and fitness-dependent mutation rates [3, 8].

Asymptotic analysis: Our result is an asymptotic runtime analysis, that is, we are interested in the runtime behavior for large problems sizes n. More formally, we view the runtime T as a function of the problem size n (even though we do not explicitly write $T(n)$) and we aim at statements on its limiting behavior. As usual in the analysis of algorithms, we use the Landau symbols $O(\cdot)$, $\Omega(\cdot)$, $\Theta(\cdot)$, $o(\cdot)$, and $\omega(\cdot)$ to conveniently describe such limits. When using such a notation, we shall always view the expression in the argument as a function of n and use the notation to describe the behavior for n tending to infinity. We note that already the algorithm parameters ε and $\rho_{\min}$ are functions of n (which is very natural since it just means that we use different parameter values for different problem sizes). Different from ε and $\rho_{\min}$, we take s as a constant (that is, not depending on n). Success rates varying with the problem size have been shown useful in [15], but generally it is much more common to have constant success rates and we do not see how non-constant success rates could be advantageous in our setting.

Since we are interested in asymptotic results only, we can and shall assume in the remainder that n is sufficiently large.

2.5 Proof of Theorem 2.1

As a first step towards understanding how our EA adjusts the mutation rate, we first determine and estimate the target mutation rate $\rho^*(\ell, s)$ introduced in the beginning of Section 2.1.

Lemma 2.3 (estimates for ρ^*). *Let $\ell \geq 1$ and $\rho^* = \rho^*(\ell, s)$. Then $\rho^* = 1 - (s+1)^{-1/\ell}$ and*

$$\frac{\ln(s+1)}{\ell}\left(1 + \frac{\ln(s+1)}{\ell}\right)^{-1} \leq \rho^* \leq \frac{\ln(s+1)}{\ell}.$$

Consequently, $\rho^* = \Theta(1/\ell)$ and $\rho^* \leq \rho^*(1, s) < 1$ is bounded away from 1 by at least a constant. If $\ell = \omega(1)$, then $\rho^* = (1 - o(1)) \ln(s + 1)/\ell$.

We now show that the success probability $p_{\text{suc}}(\rho, \ell)$, $\ell \geq 1$, changes by a factor of $(1 \mp \Omega(\delta))$ when we replace the target rate ρ^* by $\rho^*(1 \pm \delta)$. Note that for $\ell = 0$, we have $p_{\text{suc}}(\rho, \ell) = 1$ for all ρ.

Lemma 2.4 (success probabilities around ρ^*). *Let $\ell \geq 1$. Let $p_{suc}(\rho) := p_{suc}(\rho, \ell) = (1-\rho)^\ell$ for all $\rho \in [0, 1]$.*

- *For all $\delta > 0$ such that $(1 + \delta)\rho^* \leq 1$, we have*

  $p_{suc}((1+\delta)\rho^*) \leq p_{suc}(\rho^*)(1 - \frac{1}{2}\min\{\delta, \frac{1}{\ln(s+1)}\}\rho^*\ell)$.

- *For all $0 < \delta \leq 1$, we have*

  $$p_{suc}((1-\delta)\rho^*) \geq p_{suc}(\rho^*)(1 + \delta\rho^*\ell).$$

From the previous lemma we now derive that we have an at least multiplicative drift [14] towards a small interval around the target rate $\rho^*(\ell, s)$, which allows to prove upper bounds for the time to enter such an interval. For convenience, we show a bound that holds with probability $1 - 1/n$ even though we shall later only need a failure probability of $o(1)$.

Lemma 2.5. *Assume that the self-adjusting $(1 + 1)$ EA is started with a search point of fitness $\ell \geq 1$ and with initial mutation rate $\rho_0 \in [\rho_{\min}, \rho_{\max}]$, with $\rho_{\min} \in o(n^{-1}) \cap \Omega(n^{-c})$. Let $\rho^* = \rho^*(\ell, s)$. Let $\delta = \omega(\varepsilon) \cap o(1)$. For*

$$t := (1 + o(1))2\frac{\max\{\rho_0/\rho^*, \rho^*/\rho_0\}) + \ln(n)}{\delta\rho^*\ell\varepsilon} = \Theta\left(\frac{\log n}{\delta\varepsilon}\right),$$

the time T^* until a search point with higher fitness is generated or the mutation rate ρ_{T^*} is in $[(1-\delta)\rho^*, (1+\delta)\rho^*]$ satisfies $\mathbb{P}[T^* \geq t] \leq \frac{1}{n}$.

For $\ell = 0$, we have that within $\lceil \log_{1+\varepsilon}(1/\rho_0)/s \rceil + 1 = O(\log n/\varepsilon)$ iterations with probability one an improvement is found.

To ease the analysis of the mutation rate adjustment, we shall here and in a few further lemmas regard the variant of the self-adjusting $(1 + 1)$ EA which, in case it generates an improving solution, does not accept this solution, but instead continues with the parent. It is clear that the mutation rate behaves identical in this variant and in the original self-adjusting $(1 + 1)$ EA until the point when an improving solution is generated. We call this EA the *self-adjusting $(1 + 1)$ EA ignoring improvements*. This variant allows us to study the fluctuations of the mutation rate in a clean way. While the previous lemma showed that the mutation rate approaches ρ^* rapidly, the next lemma shows that once the mutation rate is close to ρ^*, with probability $1 - o(1)$ it will stay close for most of the following T rounds.

Lemma 2.6. *Let $\delta = o(1)$ be such that $\delta/\ln(1/\delta) = \omega(\varepsilon)$. There is a $\gamma = o(1)$ such that the following is true. Let $\ell \in [1..n]$, $\rho^* := \rho^*(\ell, s)$, and $\rho_0 \in [(1-\delta)\rho^*, (1+\delta)\rho^*]$. Consider a run of the self-adjusting $(1 + 1)$ EA ignoring improvements, started with a search point of fitness ℓ and with the initial mutation rate ρ_0. Denote the mutation rate after the adjustment made in iteration t by ρ_t. Then for any $T = \omega(1)$, with probability $1 - o(1)$ we have*

$$|\{t \in [1..T] \mid \rho_t \notin [(1-\gamma)\rho^*, (1+\gamma)\rho^*]\}| = o(T).$$

The last two lemmas show that the self-adjusting $(1 + 1)$ EA will likely spend most rounds with mutation rates close to ρ^*. In the next lemma we prove that in such a range the improvement probability $p_{\text{imp}}(\rho, \ell) = (1-\rho)^\ell\rho$ does not substantially decrease by the fluctuations of the mutation rate.

Lemma 2.7. *Let $\ell \in [1..n-1]$ and $\rho^* := \rho^*(\ell, s)$. Let $\gamma = o(1)$ and $\rho \in [(1-\gamma)\rho^*, (1+\gamma)\rho^*]$. Then $p_{imp}(\rho, \ell) \geq p_{imp}(\rho^*, \ell)(1 - O(\gamma))$.*

We now have the necessary prerequisites to show the main ingredient of our runtime analysis, the statement that the time to leave fitness level ℓ is (essentially) at least as good as if the EA would always use the target mutation rate $\rho^*(\ell, s)$, and this not only with respect to the expectation, but also when regarding distributions.

Lemma 2.8. *Let c be a constant and $\rho_{\min} \in o(n^{-1}) \cap \Omega(n^{-c})$. Let $\varepsilon = \omega(\log n/n) \cap o(1)$. Let $\delta = o(1)$ be such that $\delta/\ln(1/\delta) = \omega(\varepsilon)$ and $\delta = \omega(\log n/(n\varepsilon))$. Assume that the self-adjusting $(1 + 1)$ EA is started with a search point of fitness $\ell \in [0..n-1]$ and an arbitrary mutation rate $\rho \geq \rho_{\min}$. Let $\rho^* = \rho^*(\ell, s)$. Then the number T_ℓ of iterations until a search point with fitness better than ℓ is found is stochastically dominated by*

$$T_\ell \preceq o(n) + \text{Geom}(\min\{\omega(\tfrac{1}{n}), (1 - o(1))(1-\rho^*)^\ell\rho^*\}).$$

In particular, $E[T_\ell] \leq o(n) + \frac{1}{(1-\rho^*)^\ell\rho^*}$.

Having shown this bound for the time needed to leave each fitness level, we can now derive from it a bound for the whole runtime. In principle, Wegener's fitness level method [28] would be an appropriate tool here as it, essentially, states that the runtime is the sum of the times needed to leave each fitness level. For the LeadingOnes function, however, it has been observed that many algorithms visit each fitness level only with probability $\frac{1}{2}$, so by simply using the fitness level method we would lose a factor of two in the runtime guarantee. Since we believe that our runtime results are tight up to lower order terms, we care about this factor of two.

The first result in this direction is the precise runtime analysis of the $(1 + 1)$ EA with static and fitness-dependent mutation rates on LeadingOnes in [3]. The statement that the runtime is half of the sum of the exit times of the fitness levels was stated (for expected times) before Theorem 3 in [3], but a formal proof (which could easily be obtained from Theorem 2 there) was not given. We note that in parallel a second precise runtime analysis of the $(1 + 1)$ EA on LeadingOnes was given in [27] (with a conference version appearing at the same venue as [3]). Since it in particular determines the runtime precisely including the leading constant, it also cannot rely on the basic fitness level method. That the runtime is half the sum of the exit times, however, is only implicit in the computation of the expected runtime.

A more general result based on stochastic domination was presented and formally proven in [8, Theorem 3 of full version]. Unfortunately, this result was formulated only for algorithms using the same mutation operator in all iterations spent on one fitness level since this implies that the time to leave a fitness level follows a geometric distribution. This result is thus not applicable to our self-adjusting $(1+1)$ EA. By a closer inspection of the proof, we observe that the restriction to using the same mutation operator in all iterations on one fitness level is not necessary when the result is formulated via geometric distributions. We thus obtain the following result that serves our purposes.

Theorem 2.9. *Consider a $(1+1)$ EA which may use in each iteration a different unbiased mutation operator. This choice may depend on the whole history. Consider that we use this algorithm to optimize the* LeadingOnes *function. For each $\ell \in [0..n-1]$ let T_ℓ be a random variable that, regardless of how the algorithm reached this fitness level, stochastically dominates the time the algorithm takes to go from a random solution with fitness exactly ℓ to a better solution. Then the runtime T of this $(1+1)$ EA on the* LeadingOnes *function is stochastically dominated by $T \preceq \sum_{\ell=0}^{n-1} X_\ell T_\ell$, where the X_ℓ are uniformly distributed binary random variables and all X_ℓ and T_ℓ are independent. In particular, the expected runtime satisfies $E[T] \leq \frac{1}{2}\sum_{\ell=0}^{n-1} E[T_\ell]$.*

Proof of Theorem 2.1. Choose $\delta \in o(1)$ such that $\delta/\ln(1/\delta) = \omega(\varepsilon)$ and $\delta = \omega(\log n/(n\varepsilon))$. Note that such a δ exists, e.g., $\delta = \max\{\sqrt{\varepsilon}, \sqrt{\log n/(n\varepsilon)}\}$. Now Lemma 2.8 gives upper bounds for the times T_ℓ to leave the ℓ-th fitness level, which are independent of the mutation rate present when entering the fitness level. Hence by Theorem 2.9, the required stochastic dominance follows.

For the last claim (1) in Theorem 2.1, the first inequality is an immediate consequence of the domination statement. For the second one, we use the bound $\rho^* = (1-o(1))\ln(s+1)/\ell$ from Lemma 2.3. This implies in particular that for $\ell = \omega(1)$ we have $(1-\rho^*)^\ell = (1-o(1))e^{-\rho^*\ell} = (1-o(1)) \cdot 1/(s+1)$. The second step in (1) then follows by plugging in. □

3 THE SELF-ADJUSTING $(1+1)$ EA$_{>0}$

We now extend our findings for the $(1+1)$ EA to the $(1+1)$ EA$_{>0}$, which enforces that offspring are different from their parents by ensuring that at least one bit is flipped by the standard bit mutation operator. That is, the $(1+1)$ EA$_{>0}$ differs from the $(1+1)$ EA only in the choice of the mutation strength k, which in the $(1+1)$ EA follows the binomial distribution $\text{Bin}(n,p)$, and in the $(1+1)$ EA$_{>0}$ follows the conditional binomial distribution $\text{Bin}_{>0}(n,p)$ which assigns every positive integer $1 \leq m \leq n$ a probability of $\binom{n}{m}p^m(1-p)^{n-m}/(1-(1-p)^m)$. The self-adjusting version of the $(1+1)$ EA$_{>0}$ implements the same change, and can thus be obtained from Algorithm 1 by exchanging line 3 by "Sample k from $\text{Bin}_{>0}(n,p)$". This is also the algorithm empirically studied in [17].

It is clear that for static mutation rates the $(1+1)$ EA$_{>0}$ is strictly better than the plain $(1+1)$ EA, since it simply avoids the useless iterations in which duplicates of the parent are evaluated. For example, it reduces the running of the $(1+1)$ EA with static mutation rate $1/n$ by a multiplicative factor of $(e-1)/e$ [5]. For the self-adjusting $(1+1)$ EA$_{>0}$, however, it is *a priori* not evident how the conditional sampling of the mutation strengths influences the runtime. Note that after each iteration in which no bit is flipped by the $(1+1)$ EA (e.g., a $1/e$ fraction of iterations for mutation rate $1/n$), the mutation rate is increased by the factor F^s. Since these steps are avoided by the self-adjusting $(1+1)$ EA$_{>0}$, it could, in principle, happen that the actual mutation rates are smaller than what they should be. We show in this section that this is not the case. Put differently, we show that the self-adjusting $(1+1)$ EA$_{>0}$ also achieves very efficient optimization times. In contrast to the results proven in Section 2.1, however, we will obtain a bound that is difficult to evaluate in closed form. We therefore have to resort to a numerical evaluation of the proven bound. A comparison with the best possible $(1+1)$ EA$_{>0}$ variant using optimal fitness-dependent mutation rates will show that the obtained runtimes are very similar, cf. Section 3.1.

Before we can state the main theorem of this section, Theorem 3.1, we first need to discuss how the conditional sampling of the mutation strengths influences the improvement and the success probabilities. It is not difficult to see that the *improvement probability* $\hat{p}_{\text{imp}}(\rho,\ell)$ of the $(1+1)$ EA$_{>0}$, started in an arbitrary search point x with $\text{Lo}(x) = \ell$ and using mutation rate ρ, equals

$$\hat{p}_{\text{imp}}(\rho,\ell) = \frac{(1-\rho)^\ell \rho}{1-(1-\rho)^n}, \tag{2}$$

which is the improvement probability of the $(1+1)$ EA divided by the probability that the unconditional standard bit mutation creates a copy of its input.

Likewise, the *success probability* $\hat{p}_{\text{suc}}(\rho,\ell)$ of the $(1+1)$ EA$_{>0}$ in the same situation can be computed as

$$\hat{p}_{\text{suc}}(\rho,\ell) = \frac{(1-\rho)^\ell(1-(1-\rho)^{n-\ell})}{1-(1-\rho)^n} = 1 - \frac{1-(1-\rho)^\ell}{1-(1-\rho)^n}, \tag{3}$$

where the probability in the numerator is given by the probability of not flipping one of the first ℓ bits times the probability to flip at least one bit in the last $n-\ell$ positions.

As we did for the $(1+1)$ EA, we would like to define for the $(1+1)$ EA$_{>0}$ a target mutation rate $\hat{\rho}^*(\ell,s)$ to be the one that guarantees that the success probability equals $1/(s+1)$, i.e., as the value of $\hat{\rho}^*$ that solves the equation

$$\hat{p}_{\text{suc}}(\hat{\rho}^*,\ell) = 1/(s+1). \tag{4}$$

However, while the corresponding equation for the $(1+1)$ EA has always a (unique) solution, Equation (4) has a solution only if $\ell < sn/(s+1)$ (proof omitted). For this range, our analysis follows closely the one for $(1+1)$ EA, except that the algebra gets considerably more involved. I.e., Equation (4) defines a *target mutation rate* $\hat{\rho}^*(\ell,s)$, the real mutation rate ρ approaches $\hat{\rho}^*(\ell,s)$ quickly, and stays close to $\hat{\rho}^*(\ell,s)$ until a new level is reached. For larger ℓ, the mutation rate ρ has always a negative drift, and thus quickly reaches values $o(1/n)$. In this regime, the $(1+1)$ EA$_{>0}$ mimics RLS, which flips exactly one bit in each round. For technical reasons, we will set the threshold between the two regimes not at $\ell = sn/(s+1)$, but at a slightly smaller value ℓ_0. This trick helps to avoid some border cases. More precisely, throughout this section we fix $\eta_0 > 0$ and $\ell_0 \in [0,n]$ such that

$$\ell_0 := (1-\eta_0)\frac{sn}{s+1}, \qquad \text{where} \qquad \eta_0 = o(1) \tag{5}$$

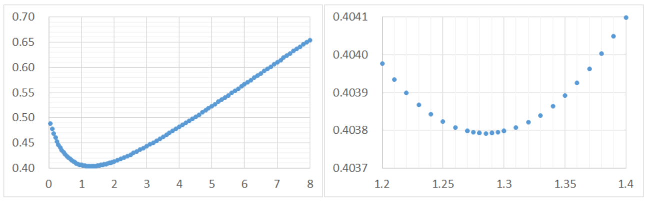

Figure 2: By n^2 normalized optimization times of the self-adjusting $(1+1)$ EA$_{>0}$ on the 10 000-dimensional LeadingOnes function for different success ratios s.

With these preparations, the main result can be stated as follows.

Theorem 3.1. *Let $c > 1$ be a constant. Consider a run of the self-adjusting $(1+1)$ EA$_{>0}$ with $F = 1 + \varepsilon$, where $\varepsilon \in \omega(\log n/n) \cap o(1)$, and with $\rho_{\min} \in o(n^{-1}) \cap \Omega(n^{-c})$, $\rho_{\max} = 1$, and arbitrary initial rate $\rho_0 \in [\rho_{\min}, \rho_{\max}]$, on the n-dimensional LeadingOnes function. Let $\eta_0 := \max\{\varepsilon^{1/6}, (\varepsilon n/\log n)^{-1/2}\}$, and let $\ell_0 := \lfloor (1-\eta_0)sn/(s+1)\rfloor$. Then the number T of iterations until the optimum is found is stochastically dominated by*

$$\sum_{\ell=0}^{\ell_0} X_\ell \operatorname{Geom}\left(\min\left\{\omega(\tfrac{1}{n}), (1-o(1))\frac{1-(1-\hat{\rho}^*(\ell,s))^n}{(1-\hat{\rho}^*(\ell,s))^\ell \hat{\rho}^*(\ell,s)}\right\}\right)$$
$$+ \sum_{\ell=\ell_0+1}^{n} X_\ell \operatorname{Geom}((1-o(1))/n) + o(n^2),$$

where the X_ℓ are uniformly distributed binary random variables and all X_ℓ and geometric random variables are mutually independent. Further, all asymptotic notation solely is with respect to n and can be chosen uniformly for all ℓ. In particular,

$$E[T] \le (1+o(1))\frac{1}{2}\left(\frac{n^2}{s+1} + \sum_{\ell=0}^{\ell_0} \frac{1-(1-\hat{\rho}^*(\ell,s))^n}{(1-\hat{\rho}^*(\ell,s))^\ell \hat{\rho}^*(\ell,s)}\right). \qquad (6)$$

3.1 Numerical Evaluation

As mentioned above, the evaluation of the runtime bound (6) is not as straightforward as the corresponding one of the unconditional $(1+1)$ EA. For a proper evaluation, one would have to compute bounds on $\hat{\rho}^*(\ell,s)$, and then plug these into the runtime bound. Since these computations are quite tedious, we will content ourselves by numerically solving (4) for $\hat{\rho}^*(\ell,s)$ with Mathematica™, and obtaining a numerical approximation for the runtime.

Before estimating $\mathbb{E}[T]$, we briefly discuss the $(1+1)$ EA$_{>0,\text{opt}}$, the $(1+1)$ EA$_{>0}$ variant that uses in each round the mutation rate $p_{>0,\text{opt}}(\text{Lo}(x))$ which maximizes the improvement probability (2). The performance of this algorithm is a lower bound for the performance of any $(1+1)$ EA$_{>0}$ variant, and thus for our self-adjusting $(1+1)$ EA$_{>0}$. We again do not compute $p_{>0,\text{opt}}(\ell)$ exactly, but only numerically. For $n \in \{100, 1\,000, 10\,000\}$ and all $0 \le \ell < n/2$, the numerically computed values are quite close, but not identical to $1/(\ell+1)$, which was the optimal rate for the $(1+1)$ EA. For $n = 10\,000$ the largest difference between $p_{>0,\text{opt}}(\ell)$ and $1/(\ell+1)$ is 0.0001741 and the smallest is -0.0000382. For $\ell \ge n/2$, it is not difficult to see that $p_{>0,\text{opt}}(\ell)$ is obtained by the limit at 0, in which case the $(1+1)$ EA$_{>0}$ reduces to RLS. The expected runtime of $(1+1)$ EA$_{>0,\text{opt}}$ is

$$1 + \frac{1}{2}\sum_{\ell=0}^{n-1} \min\left\{n, \frac{1-(1-p_{>0,\text{opt}}(\ell))^n}{p_{>0,\text{opt}}(\ell)(1-p_{>0,\text{opt}}(\ell))^\ell}\right\}$$

For $n = 100$ ($n = 1\,000$, $n = 10\,000$) this expression evaluates to approximately 0.4077 (0.4026, 0.4027) when normalized by n^2. As a side remark, we note that the expected runtime of the best possible unary unbiased algorithm for these problem dimensions has normalized runtime of around 0.3884. The $(1+1)$ EA$_{>0,\text{opt}}$ is thus only around 3.7% worse than this RLS$_{\text{opt}}$ heuristic. Put differently, the cost of choosing the mutation rates from $\text{Bin}_{>0}(n,p)$ instead of deterministically using the optimal fitness-dependent mutation strength is only 3.7%. For comparison, we recall that the (unconditional) $(1+1)$ EA variant using optimal mutation rates has an expected normalized runtime of $e/4 \approx 0.6796$, which is about 75% worse than that of RLS$_{\text{opt}}$.

We now estimate how close the performance of the self-adjusting $(1+1)$ EA$_{>0}$ gets to this $(1+1)$ EA$_{>0,\text{opt}}$. To this end, we fix $n = 10\,000$ and compute $\hat{\rho}^*(\ell,s)$ for different success ratios s. The normalized expected runtimes are plotted in Fig. 2. The interesting region of success ratios between 1.2 and 1.4 is plotted in the zoom on the right. For this n, the best success ratio is around 1.285, which gives a normalized expected runtime of around 0.403792. This value is only 0.26% larger than the expected runtime of the $(1+1)$ EA$_{>0,\text{opt}}$ for $n = 10\,000$. A numerical evaluation for $n = 50\,000$ shows that the optimal success rate is again around 1.285, giving a normalized expected runtime slightly less than 0.40375375.

3.2 Proof Overview

The proof of Theorem 3.1 relies on the same techniques as the proof for the $(1+1)$ EA, but it needs to reflect the different regimes $\ell \le \ell_0$ and $\ell > \ell_0$, where $\ell_0 = (1-\eta_0)sn/(s+1)$. We only give an outline.

For $\ell \le \ell_0$, the strategy is very similar to the $(1+1)$ EA, but the asymptotic analysis becomes more involved. For example, when $\ell < (1-\Omega(1))sn/(s+1)$ then $\hat{\rho}^* = \Theta(1/\ell)$, but if $\ell = (1-\eta)sn/(s+1)$ for some $\eta = o(1)$ then $\hat{\rho}^* = \Theta(\eta/\ell) = o(1/\ell)$. Thus the order of $\hat{\rho}^*$ depends on whether ℓ is far from ℓ_0, or close to ℓ_0. For the latter regime, if we want to use (3) to compute the success probability $\hat{p}_{\text{suc}}(\rho,\ell)$ for slightly varying values of ρ then we need to be very precise. Numerator and denominator of (3) vary by the same first-order error terms, and it is the second-order error term which actually dominates the fluctuation of $\hat{p}_{\text{suc}}(\rho,\ell)$. Nevertheless, we can show that if $\rho \approx \hat{\rho}^*$ varies by a factor of $(1 \pm \delta)$ then $\hat{p}_{\text{suc}}(\rho,\ell)$ varies by at least a factor of $1 \mp \Omega(\delta\hat{\rho}^*\ell)$ respectively, just as in Lemma 2.4 for the $(1+1)$ EA. Thus we can use the same ideas as for the $(1+1)$ EA.

For $\ell \ge \ell_0$, the situation is a bit easier. We simply show that for all mutation rates $\rho \ge \Omega(\eta_0/n)$, for a suitable hidden constant, the success probability is bounded away from $1/(s+1)$, and thus the mutation rate has strong drift towards zero. Hence, the mutation rate spends most of the time below $O(\eta_0/n) = o(1/n)$, and in this regime the $(1+1)$ EA$_{>0}$ resembles RLS. Indeed, using the same ideas as before it is not hard to show that the time to leave a fitness level is dominated by $o(n) + \text{Geom}((1-o(1))/n)$, as we would expect for the RLS. We omit the details.

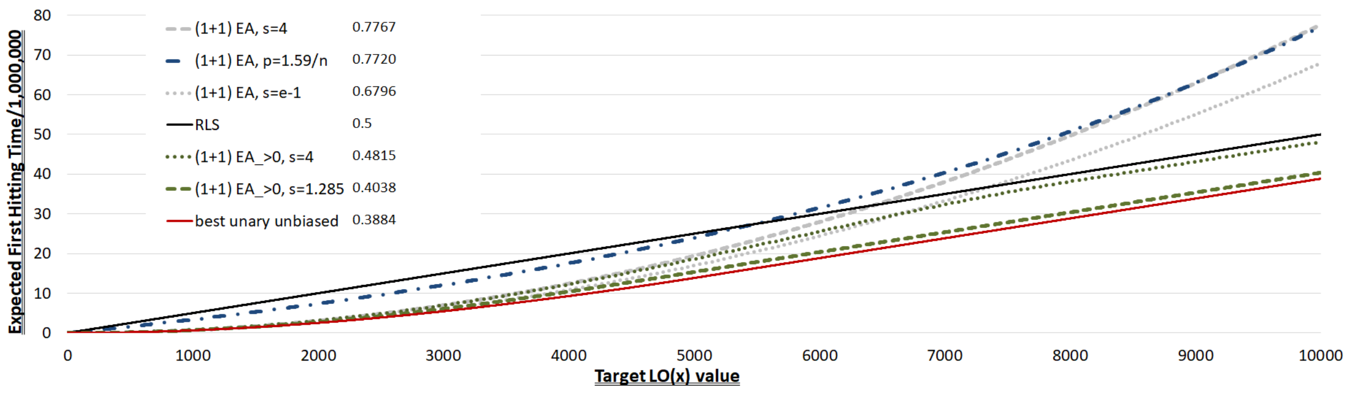

Figure 3: Expected fixed target runtimes for LeadingOnes in dimension $n = 10\,000$. The curve of $(1+1)$ EA$_{>0,\text{opt}}$ is not plotted since it is indistinguishable from that of the $(1+1)$ EA$_{>0}$ with success ratio $s = 1.285$ and the curve of the $(1+1)$ EA$_{\text{opt}}$ is not plotted since it is indistinguishable from that of the $(1+1)$ EA with success ratio $s = e - 1$. The values shown in the legend are the by n^2 normalized expected optimization times.

4 FIXED TARGET RUNTIMES

Our main focus in the previous sections, and in particular in the sections presenting numerical evaluations of the self-adjusting $(1+1)$ EA variants (i.e., Sections 2.2 and 3.1), was on computing the expected optimization time. We now follow a suggestion previously made in [4], and study the anytime performance of the algorithms, by analyzing their expected fixed-target runtimes. That is, for an algorithm A we regard for each target value $0 \leq v \leq n$ the expected number $\mathbb{E}[T(n, A, v)]$ of function evaluations until algorithm A evaluates for the first time a solution x which satisfies $\text{Lo}(x) \geq v$.

Fig. 3 plots these expected fixed target runtimes of selected algorithms for $n = 10\,000$. The legend also mentions the expected overall optimization time, i.e., $\mathbb{E}[T(n, A, 10\,000)]$. We do not plot the $(1+1)$ EA$_{>0,\text{opt}}$, since its runtime would be indistinguishable in this plot from the self-adjusting $(1+1)$ EA$_{>0}$ with success ratio $s = 1.285$. For the same reason we do not plot $(1+1)$ EA$_{\text{opt}}$, the $(1+1)$ EA with optimal fitness-dependent mutation rate $p = n/(\ell + 1)$, whose data is almost identical to that of the self-adjusting $(1+1)$ EA with the optimal success ratio $s = e - 1$.

We plot in Fig. 3 the $(1+1)$ EA with one-fifth success rule (i.e., success ratio $s = 4$). While its overall runtime is the worst of all algorithms plotted in this figure, we see that its fixed-target runtime is better than that for RLS for all targets up to 6 436. Its overall runtime is very close to that of the $(1+1)$ EA with the best static mutation rate $p \approx 1.59/n$ [3], and for all targets $v \leq 9\,017$ the expected runtime is smaller.

We already discussed that the expected optimization time of the two algorithms $(1+1)$ EA$_{\text{opt}}$ and the self-adjusting $(1+1)$ EA with success ratio $s = e - 1$ is around 36% worse than that of RLS. However, we also see that their fixed-target performances are better for all targets up to $v = 7\,357$. For example, for $v = 5\,000$ their expected first hitting time is slightly less than $17 * 10^6$ and thus about 36% smaller than that of RLS.

As we have seen already in Fig. 2, the self-adjusting $(1+1)$ EA$_{>0}$ with success ratio $s = 4$ (i.e., using a one-fifth success rule) has an overall runtime similar, but slightly better than RLS. We recall that its target mutation rate is 0 for values $v \geq 4n/5$. In this regime the slope of its fixed target runtime curve is thus identical to that of RLS. For the self-adjusting $(1+1)$ EA$_{>0}$ this is the case for v slightly larger than 5 600. The $(1+1)$ EA$_{>0,\text{opt}}$ with optimal fitness-dependent mutation rate uses mutation rate $p = 0$ for $v \geq 4\,809$.

We also observe that the best unary unbiased black-box algorithm for LeadingOnes, which is an RLS-variant with fitness-dependent mutation strength (cf. [8, 17] for more detailed discussions), is also best possible for all intermediate targets $v < 0$. It is not difficult to verify this formally, the main argument being that the fitness landscape of LeadingOnes is non-deceptive.

5 CONCLUSIONS

We have proven upper bounds for the $(1+1)$ EA and $(1+1)$ EA$_{>0}$ with success-based multiplicative update rules using constant success ratio s and update strengths $F = 1 + o(1)$. In particular, we have shown that the $(1+1)$ EA with $1/e$-th success rule achieves asymptotically optimal runtime (for update strengths $F = 1 + o(1)$). For the $(1+1)$ EA$_{>0}$, numerical evaluations for $n = 10\,000$ and $n = 50\,000$ suggest a success ratio of around 1.285; with this success rate the self-adjusting $(1+1)$ EA$_{>0}$ achieves an expected runtime around $0.40375n^2 + o(n^2)$. Our precise upper bounds are stochastic domination bounds, which allow to derive other moments of the runtime.

Our work continues a series of recent papers rigorously demonstrating advantages of controlling the parameters of iterative heuristics during the optimization process. Developing a solid understanding of problems for which simple success-based update schemes are efficient, and which problems require more complex control mechanisms (e.g., based on reinforcement learning [13], or techniques using statistics for the success rate within a window of iterations [15, 24]) is the long-term goal of our research.

ACKNOWLEDGMENTS

Our work was supported by a public grant as part of the Investissement d'avenir project, reference ANR-11-LABX-0056-LMH, and by the Paris Ile-de-France Region.

REFERENCES

[1] Aldeida Aleti and Irene Moser. 2016. A Systematic Literature Review of Adaptive Parameter Control Methods for Evolutionary Algorithms. *Comput. Surveys* 49 (2016), 56:1–56:35.

[2] Anne Auger. 2009. Benchmarking the (1+1) evolution strategy with one-fifth success rule on the BBOB-2009 function testbed. In *Companion Material for Proc. of Genetic and Evolutionary Computation Conference (GECCO'09)*. ACM, 2447–2452.

[3] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. 2010. Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem. In *Proc. of Parallel Problem Solving from Nature (PPSN'10) (Lecture Notes in Computer Science)*, Vol. 6238. Springer, 1–10.

[4] Eduardo Carvalho Pinto and Carola Doerr. 2017. Discussion of a More Practice-Aware Runtime Analysis for Evolutionary Algorithms. In *Proc. of Artificial Evolution (EA'17)*. 298–305. Extended version available online at https://arxiv.org/abs/1812.00493.

[5] Eduardo Carvalho Pinto and Carola Doerr. 2018. A Simple Proof for the Usefulness of Crossover in Black-Box Optimization. In *Proc. of Parallel Problem Solving from Nature (PPSN'18) (Lecture Notes in Computer Science)*, Vol. 11102. Springer, 29–41.

[6] Luís Da Costa, Álvaro Fialho, Marc Schoenauer, and Michèle Sebag. 2008. Adaptive operator selection with dynamic multi-armed bandits. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'08)*. ACM, 913–920.

[7] Luc Devroye. 1972. *The compound random search.* Ph.D. dissertation, Purdue Univ., West Lafayette, IN.

[8] Benjamin Doerr. 2018. Better Runtime Guarantees via Stochastic Domination. In *Proc. of Evolutionary Computation in Combinatorial Optimization (EvoCOP'18) (Lecture Notes in Computer Science)*, Vol. 10782. Springer, 1–17. Full version available at http://arxiv.org/abs/1801.04487.

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

[10] Benjamin Doerr and Carola Doerr. 2018. Theory of Parameter Control Mechanisms for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices. In *Theory of Randomized Search Heuristics in Discrete Search Spaces*, Benjamin Doerr and Frank Neumann (Eds.). Springer. To appear. Available online at https://arxiv.org/abs/1804.05650.

[11] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From black-box complexity to designing new genetic algorithms. *Theoretical Computer Science* 567 (2015), 87–104.

[12] Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2019. Self-Adjusting Mutation Rates with Provably Optimal Success Rules. *CoRR* abs/1902.02588 (2019). arXiv:1902.02588 A GitHub repository with numerical evaluations is available at https://github.com/CarolaDoerr/2019-LO-SelfAdjusting.

[13] Benjamin Doerr, Carola Doerr, and Jing Yang. 2016. k-Bit Mutation with Self-Adjusting k Outperforms Standard Bit Mutation. In *Proc. of Parallel Problem Solving from Nature (PPSN'16) (Lecture Notes in Computer Science)*, Vol. 9921. Springer, 824–834.

[14] Benjamin Doerr, Daniel Johannsen, and Carola Winzen. 2012. Multiplicative Drift Analysis. *Algorithmica* 64 (2012), 673–697.

[15] Benjamin Doerr, Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. 2018. On the Runtime Analysis of Selection Hyper-Heuristics with Adaptive Learning Periods. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 1015–1022.

[16] Benjamin Doerr, Carsten Witt, and Jing Yang. 2018. Runtime Analysis for Self-adaptive Mutation Rates. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 1475–1482.

[17] Carola Doerr and Markus Wagner. 2018. On the Effectiveness of Simple Success-Based Parameter Selection Mechanisms for Two Classical Discrete Black-Box Optimization Benchmark Problems. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'18)*. ACM, 943–950.

[18] Álvaro Fialho, Luís Da Costa, Marc Schoenauer, and Michèle Sebag. 2010. Analyzing bandit-based adaptive operator selection mechanisms. *Annals of Mathematics and Artificial Intelligence* 60 (2010), 25–64.

[19] Thomas Jansen, Kenneth A. De Jong, and Ingo Wegener. 2005. On the Choice of the Offspring Population Size in Evolutionary Algorithms. *Evolutionary Computation* 13 (2005), 413–440.

[20] Giorgos Karafotias, Ágoston E. Eiben, and Mark Hoogendoorn. 2014. Generic parameter control with reinforcement learning. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'14)*. ACM, 1319–1326.

[21] Giorgos Karafotias, Mark Hoogendoorn, and A.E. Eiben. 2015. Parameter Control in Evolutionary Algorithms: Trends and Challenges. *IEEE Transactions on Evolutionary Computation* 19 (2015), 167–187.

[22] Stefan Kern, Sibylle D. Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. 2004. Learning probability distributions in continuous evolutionary algorithms - a comparative review. *Natural Computing* 3 (2004), 77–112.

[23] Jörg Lässig and Dirk Sudholt. 2014. General Upper Bounds on the Runtime of Parallel Evolutionary Algorithms. *Evolutionary Computation* 22 (2014), 405–437.

[24] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. 2017. On the Runtime Analysis of Generalised Selection Hyper-heuristics for Pseudo-Boolean Optimisation. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'17)*. ACM, 849–856. Extended version available at https://arxiv.org/abs/1801.07546.

[25] Ingo Rechenberg. 1973. *Evolutionsstrategie.* Friedrich Fromman Verlag (Günther Holzboog KG), Stuttgart.

[26] Michael A. Schumer and Kenneth Steiglitz. 1968. Adaptive step size random search. *IEEE Trans. Automat. Control* 13 (1968), 270–276.

[27] Dirk Sudholt. 2012. Crossover speeds up building-block assembly. In *Proc. of Genetic and Evolutionary Computation Conference (GECCO'12)*. ACM, 689–702.

[28] Ingo Wegener. 2001. Theoretical Aspects of Evolutionary Algorithms. In *Proc. of the 28th International Colloquium on Automata, Languages and Programming (ICALP'01) (Lecture Notes in Computer Science)*, Fernando Orejas, Paul G. Spirakis, and Jan van Leeuwen (Eds.), Vol. 2076. Springer, 64–78.