Self-Adjusting Population Sizes for Non-Elitist Evolutionary Algorithms: Why Success Rates Matter*

Mario Alejandro Hevia Fajardo
Department of Computer Science
University of Sheffield, Sheffield, UK

Dirk Sudholt
Chair of Algorithms for Intelligent Systems
University of Passau, Passau, Germany

ABSTRACT

Recent theoretical studies have shown that self-adjusting mechanisms can provably outperform the best static parameters in evolutionary algorithms on discrete problems. However, the majority of these studies concerned elitist algorithms and we do not have a clear answer on whether the same mechanisms can be applied for non-elitist algorithms.

We study one of the best-known parameter control mechanisms, the one-fifth success rule, to control the offspring population size λ in the non-elitist $(1, \lambda)$ EA. It is known that the $(1, \lambda)$ EA has a sharp threshold with respect to the choice of λ where the runtime on OneMax changes from polynomial to exponential time. Hence, it is not clear whether parameter control mechanisms are able to find and maintain suitable values of λ.

We show that the answer crucially depends on the success rate s (i. e. a one-$(s+1)$-th success rule). We prove that, if the success rate is appropriately small, the self-adjusting $(1, \lambda)$ EA optimises OneMax in $O(n)$ expected generations and $O(n \log n)$ expected evaluations. A small success rate is crucial: we also show that if the success rate is too large, the algorithm has an exponential runtime on OneMax.

CCS CONCEPTS

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

KEYWORDS

Parameter control, Theory, Runtime analysis, Non-elitism, Drift analysis, Evolutionary algorithms

ACM Reference Format:
Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2021. Self-Adjusting Population Sizes for Non-Elitist Evolutionary Algorithms: Why Success Rates Matter. In *2021 Genetic and Evolutionary Computation Conference (GECCO '21), July 10–14, 2021, Lille, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3449639.3459338

*This research has been supported by CONACYT under the grant no. 739621 and registration no. 843375.

1 INTRODUCTION

Parameter control mechanisms aim to identify parameter values that are optimal for the current state of the optimisation process. In continuous optimisation, parameter control is indispensable to ensure convergence to the optimum, therefore, non-static parameter choices have been standard for several decades. In contrast, in the discrete domain parameter control has only become more common in recent years. This is in part thanks to theoretical studies demonstrating that fitness-dependent parameter control mechanisms can provably outperform the best static parameter settings [1, 2, 6, 9]. Despite the proven advantages, fitness-dependent mechanisms have an important drawback: to have an optimal performance they generally need to be tailored to a specific problem which needs a substantial knowledge of the problem in hand [5].

To overcome this constraint, several parameter control mechanisms have been proposed that update the parameters in a *self-adjusting* manner. Theoretical studies have proven that in spite of their simplicity, these mechanisms are able to use *good* parameter values throughout the optimisation, obtaining the same or better performance than any static parameter choice.

There is a growing body of research in this rapidly emerging area. Lässig and Sudholt [18] presented self-adjusting schemes for choosing the offspring population size in $(1+\lambda)$ EAs and the number of islands in an island model. Mambrini and Sudholt [25] adapted the migration interval in island models and showed that adaptation can reduce the communication effort beyond the best possible fixed parameter. Doerr and Doerr [4] proposed a self-adjusting mechanism in the $(1 + (\lambda, \lambda))$ GA based on the one-fifth rule and proved that it optimises OneMax in $O(n)$ evaluations, being the fastest known unbiased genetic algorithm on OneMax. Hevia Fajardo and Sudholt [14] studied modifications to the self-adjusting mechanism in the $(1 + (\lambda, \lambda))$ GA on Jump functions, showing that they can perform nearly as well as the $(1 + 1)$ EA with the optimal mutation rate. Doerr, Doerr, and Kötzing [7] presented a success-based choice of the mutation strength for an RLS variant, proving that it is very efficient for a multivariate generalisation of the OneMax problem. Doerr, Gießen, Witt, and Yang [10] showed that a success-based parameter control mechanism is able to identify and track the optimal mutation rate in the $(1+\lambda)$ EA on OneMax, matching the performance of the best known fitness-dependent parameter [1]. Doerr, Doerr, and Lengler [8] proved that a success-based parameter control mechanism based on the one-fifth rule is able to achieve an asymptotically optimal runtime on LeadingOnes. Lissovoi, Oliveto, and Warwicker [24] proposed a Generalised Random Gradient Hyper-Heuristic that uses a learning period of τ steps that can learn to adapt the neighbourhood size of Random Local Search optimally during the run on LeadingOnes. This result required the correct selection of the learning period τ; this was later solved using a self-adjusting mechanism adapting the learning period having an optimal asymptotic expected runtime on LeadingOnes [11], OneMax and Ridge [22]. Rajabi and Witt [30, 31] proposed a stagnation detection mechanism that adjusts the mutation strength. The mechanism applied to the $(1+1)$ EA and RLS_k has the same asymptotic runtime as the optimal parameter setting on Jump. Rajabi and Witt [29] used a self-adjusting asymmetric mutation that, on OneMax, gives a constant-factor speedup over asymmetric mutations [16]. Doerr and Doerr give a comprehensive survey of theoretical results [5].

Most theoretical analyses of parameter control mechanisms focus on elitist algorithms (with notable exceptions that study self-adaptive mutation rate in the $(1, \lambda)$ EA [12] and the (μ, λ) EA [3], and hyper-heuristics that select between elitist and non-elitist selection [23]). Hence, the performance of parameter control mechanisms in non-elitist algorithms is not well understood. There are many applications of non-elitist evolutionary algorithms for which an improved theoretical understanding of parameter control mechanisms could bring performance improvements similar to the ones seen for elitist algorithms.

We consider the $(1, \lambda)$ EA on OneMax that in every generation creates λ offspring and selects the best one for survival. Rowe and Sudholt [32] have shown that there is a sharp threshold at $\lambda = \log_{\frac{e}{e-1}} n$ between exponential and polynomial runtimes on OneMax. A value $\lambda \geq \log_{\frac{e}{e-1}} n$ ensures that the offspring population size is sufficiently large to ensure a positive *drift* (expected progress) towards the optimum even on the most challenging fitness levels. For easier fitness levels, smaller values of λ are sufficient.

This is a challenging scenario for self-adjusting the offspring population size λ since too small values of λ can easily make the algorithm decrease its current fitness, moving away from the optimum. For static values of $\lambda \leq (1-\varepsilon) \log_{\frac{e}{e-1}} n$, for any constant $\varepsilon > 0$, we know that the optimisation time is exponential with high probability [32]. Moreover, too large values for λ can waste function evaluations and blow up the optimisation time.

We consider a self-adjusting version of the $(1, \lambda)$ EA called $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA (self-adjusting $(1, \lambda)$ EA) that uses a success-based rule. For an update strength F and a success rate s, in a generation where no improvement in fitness is found, λ is increased by a factor of $F^{1/s}$ and in a successful generation, λ is divided by a factor F. If one out of $s+1$ generations is successful, the value of λ is maintained. The case $s = 4$ is the famous one-fifth success rule.

We ask whether the self-adjusting $(1, \lambda)$ EA is able to find and maintain suitable parameter values of λ throughout the run, despite the lack of elitism and without knowledge of the problem in hand.

We answer this question in the affirmative if the success rate s is chosen correctly. We show in Section 3 that, if s is a constant with $0 < s < \frac{e-1}{e}$, then the self-adjusting $(1, \lambda)$ EA optimises OneMax in $O(n)$ expected generations and $O(n \log n)$ expected fitness evaluations. The bound on evaluations is optimal for all unary unbiased black-box algorithms [9, 19]. However, if s is a sufficiently large constant, $s \geq 22$, the runtime becomes exponential with overwhelming probability (see Section 4). The reason is that then unsuccessful generations increase λ only slowly, whereas successful generations decrease λ significantly. We show that then the algorithm gets stuck in a non-stable equilibrium with small λ-values and frequent fallbacks (fitness decreases) at a linear distance to the optimum.

To bound the expected number of generations for small success rates, we apply drift analysis to a potential function that trades off increases in fitness against a penalty term for small λ-values. In generations where the fitness decreases, λ increases and the penalty is reduced, allowing us to show a positive drift in the potential for all fitness levels and all λ.

To bound the expected number of evaluations, we further use the potential to construct a novel "ratchet argument": we show that, even when the fitness decreases, it does not decrease much below the best fitness seen so far. More precisely, with high probability, if $f(x_t)$ is the current fitness at time t and $f_t^* = \max\{f(x_{t'}) \mid t' \leq t\}$ is the best fitness seen so far, then $f(x_t) \geq f_t^* - r \log n$ for an appropriate constant r. Then we show that there is a constant probability that the best-so far fitness is increased by $\log n$ in a sequence of generations without fallbacks. We are hopeful that these arguments will prove useful in the analysis of other non-elitist algorithms as well.

Some proofs are omitted or sketched; for a full version see [15].

2 PRELIMINARIES

We study the expected number of generations and fitness evaluations of the self-adjusting $(1, \lambda)$ EA with self-adjusted offspring population size λ to find the optimum of $\text{OneMax}(x) := \sum_{i=1}^{n} x_i$. We define $X_0, X_1, \ldots$ as the sequence of states of the algorithm, where $X_t = (x_t, \lambda_t)$ describes the current search point x_t and the offspring population size λ_t at generation t. We often omit the subscripts t when the context is obvious.

Using the naming convention from [5] we call the algorithm self-adjusting $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA (Algorithm 1). The algorithm behaves as the conventional $(1, \lambda)$ EA, but in every generation it adjusts the offspring population size depending on the success of the generation. If the fittest offspring y is better than the parent x, the offspring population size is divided by the *update strength* F, and multiplied by $F^{1/s}$ otherwise, with s being the *success rate*.

The idea of the parameter control mechanism is based on the interpretation of the one-fifth success rule from [17]. The parameter λ remains constant if the algorithm has a success every $s+1$ generations as then its new value is $\lambda \cdot (F^{1/s})^s \cdot 1/F = \lambda$. In pseudo-Boolean optimisation, the one-fifth success rule was first implemented by Doerr *et al.* [6], and proved to track the optimal offspring population size on the $(1 + (\lambda, \lambda))$ GA in [4]. Our implementation is closer to the one used in [8], where the authors generalise the success rule, implementing the success rate s as a hyper-parameter.

Note that we regard λ to be a real value, so that changes by factors of $1/F$ or $F^{1/s}$ happen on a continuous scale. Following Doerr and Doerr [4], we assume that, whenever an integer value of λ is required, λ is rounded to a nearest integer. For the sake of readability, we often write λ as a real value even when an integer is required.

We now give notation and tools for all $(1, \lambda)$ EA algorithms.

Definition 2.1. For all $\lambda \in \mathbb{N}$ and $0 \leq i < n$ we define:

$$p_{i,\lambda}^{-} = \Pr(f(x_{t+1}) < i \mid f(x_t) = i)$$
$$p_{i,\lambda}^{0} = \Pr(f(x_{t+1}) = i \mid f(x_t) = i)$$
$$p_{i,\lambda}^{+} = \Pr(f(x_{t+1}) > i \mid f(x_t) = i)$$
$$\Delta_{i,\lambda}^{-} = \mathrm{E}(i - f(x_{t+1}) \mid f(x_t) = i \text{ and } f(x_{t+1}) < i)$$
$$\Delta_{i,\lambda}^{+} = \mathrm{E}(f(x_{t+1}) - i \mid f(x_t) = i \text{ and } f(x_{t+1}) > i)$$

Algorithm 1: Self-adjusting $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA.

```
1 Initialization: Choose $x \in \{0,1\}^n$ uniformly at random
                  (u.a.r.) and set $\lambda := 1$;
2 Optimization: for $t \in \{1, 2, \dots\}$ do
3     Mutation: for $i \in \{1, \dots, \lambda\}$ do
4         $y'_i \in \{0,1\}^n \leftarrow$ mutate($x$);
5     Selection: Choose $y \in \{y'_1, \dots, y'_\lambda\}$ with
                 $f(y) = \max\{f(y'_1), \dots, f(y'_\lambda)\}$ u.a.r.;
6     Update:
7     if $f(y) > f(x)$ then $x \leftarrow y; \lambda \leftarrow \max\{1, \lambda/F\}$;
8     else $x \leftarrow y; \lambda \leftarrow F^{1/s}\lambda$;
```

As in [32], we call $\Delta^+_{i,\lambda}$ *forward drift* and $\Delta^-_{i,\lambda}$ *backward drift* and note that they are both at least 1 by definition. Now, $p^+_{i,1}$ is the probability of one offspring finding a better fitness value and $p^+_{i,\lambda} = 1 - (1 - p^+_{i,1})^\lambda$ since it is sufficient that one offspring improves the fitness. The probability of a fallback is $p^-_{i,\lambda} = (p^-_{i,1})^\lambda$ since all offspring must have worse fitness than their parent. Along with common bounds $\frac{n-i}{en} \le p^+_{i,1} \le \frac{n-i}{n}$ and standard arguments, we obtain the following.

Lemma 2.2. *For any $(1, \lambda)$ EA on OneMax, the quantities from Definition 2.1 are bounded as follows.*

$$1 - \frac{en}{en + \lambda(n-i)} \le 1 - \left(1 - \frac{n-i}{en}\right)^\lambda \le p^+_{i,\lambda} \le 1 - \left(1 - \frac{n-i}{n}\right)^\lambda \quad (1)$$

$$\left(\frac{i}{n} - \frac{1}{e}\right)^\lambda \le p^-_{i,\lambda} \le \left(\frac{e-1}{e}\right)^\lambda \quad (2)$$

$$1 \le \Delta^-_{i,\lambda} \le \frac{e}{e-1} \quad (3)$$

$$1 \le \Delta^+_{i,\lambda} \le \sum_{j=1}^{\infty} \left(1 - \left(1 - \frac{1}{j!}\right)^\lambda\right) \quad (4)$$

If $\lambda \ge 5$, then $\Delta^+_{i,\lambda} \le \lceil \log \lambda \rceil + 0.413$.

3 SMALL SUCCESS RATES ARE EFFICIENT

We show that, for suitable choices of the success rate s and constant update strength F, the self-adjusting $(1, \lambda)$ EA optimises OneMax in $O(n)$ expected generations and $O(n \log n)$ expected evaluations.

3.1 Bounding the Number of Generations

Theorem 3.1. *Let the update strength $F > 1$ and the success rate $0 < s < \frac{e-1}{e}$ be constants. Then for any initial search point and any initial λ the expected number of generations of the self-adjusting $(1, \lambda)$ EA on OneMax is $O(n)$.*

As mentioned before, we make use of a potential function that captures both the fitness and the offspring population size.

Definition 3.2. We define the potential function $g(X_t)$ as

$$g(X_t) = f(x_t) - \frac{se}{e-1} \log_F \left(\max\left(\frac{enF^{1/s}}{\lambda_t}, 1\right)\right).$$

The potential function is composed of two main terms, the fitness and a penalty term that, for $\lambda_t \le en$, grows linearly in $\log_F \lambda$ (since $-\log_F \left(\frac{enF^{1/s}}{\lambda_t}\right) = -\log_F(enF^{1/s}) + \log_F(\lambda_t)$). That is, when λ increases the penalty decreases and vice-versa. The idea behind this definition is that small values of λ may lead to decreases in fitness, but these are compensated by an increase in λ and a reduction of the penalty term. For $\lambda_t \ge enF^{1/s}$ the penalty disappears as then λ is so large that improvements are likely; we will see that this is sufficient to show a positive drift in the potential.

Since the range of the penalty term is limited, the potential is always close to the current fitness as shown in the following lemma.

Lemma 3.3. *For all generations t, the fitness and the potential are related as follows: $f(x_t) - \frac{se}{e-1}\log_F(enF^{1/s}) \le g(X_t) \le f(x_t)$. In particular, $g(X_t) = n$ implies $f(x_t) = n$.*

Proof. The penalty term $\frac{se}{e-1} \log_F \left(\max\left(\frac{enF^{1/s}}{\lambda_t}, 1\right)\right)$ is a non-increasing function in λ_t with its minimum being 0 for $\lambda \ge enF^{1/s}$ and its maximum being $\frac{se}{e-1} \log_F \left(enF^{1/s}\right)$ when $\lambda = 1$. Hence, $f(x_t) - \frac{se}{e-1} \log_F(enF^{1/s}) \le g(X_t) \le f(x_t)$. □

Now we proceed to show that with the correct choice of hyper-parameters the drift in potential is at least a positive constant during all parts of the optimisation.

Lemma 3.4. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 3.1. Then for every generation t with $f(x_t) < n$,*

$$E(g(X_{t+1}) - g(X_t) \mid X_t) \ge \frac{1}{e} - \frac{s}{e-1} > 0$$

for large enough n. This also holds when only considering improvements that increase the fitness by 1.

Proof. We first consider the case $\lambda_t \le en$ as then $\lambda_{t+1} \le enF^{1/s}$ and $g(X_{t+1}) = f(x_{t+1}) - \frac{se}{e-1}(\log_F(enF^{1/s}) - \log_F(\lambda_{t+1}))$.

When an improvement is found, the fitness increases by at least $\Delta^+_{i,\lambda}$ and since $\lambda_{t+1} = \lambda_t/F$, the penalty term $\frac{se}{e-1}(\log_F(enF^{1/s}) - \log_F(\lambda_t))$ increases by $\frac{se}{e-1}$ (unless $\lambda_{t+1} = 1$ is reached, in which case the increase might be lower). When the fitness does not change, the penalty decreases by $\frac{e}{e-1}$. When the fitness decreases, the expected decrease is at most $\Delta^-_{i,\lambda}$ and the penalty decreases by $\frac{e}{e-1}$. Together, $\mathrm{E}\left(g(X_{t+1}) - g(X_t) \mid X_t, \lambda_t \le en\right)$ is at least

$$p^+_{i,\lambda}\left(\Delta^+_{i,\lambda} - \frac{se}{e-1}\right) + p^0_{i,\lambda} \cdot \frac{e}{e-1} + p^-_{i,\lambda}\left(-\Delta^-_{i,\lambda} + \frac{e}{e-1}\right).$$

Using $\Delta^+_{i,\lambda} \ge 1$ (which also holds when only considering fitness increases by 1) and $\Delta^-_{i,\lambda} \le \frac{e}{e-1}$ by Lemma 2.2, this is at least

$$p^+_{i,\lambda}\left(1 - \frac{se}{e-1}\right) + p^0_{i,\lambda} \cdot \frac{e}{e-1}.$$

We bound the second summand from below using $\frac{e}{e-1} > 1 - \frac{se}{e-1}$ (note that the left-hand side is larger than 1 and the right-hand side is less than 1) and obtain a lower bound of

$$p^+_{i,\lambda}\left(1 - \frac{se}{e-1}\right) + p^0_{i,\lambda}\left(1 - \frac{se}{e-1}\right) = (1 - p^-_{i,\lambda})\left(1 - \frac{se}{e-1}\right).$$

Lemma 2.2 shows that $p^-_{i,\lambda} \le \frac{e-1}{e}$ for all λ, hence $1 - p^-_{i,\lambda} \ge \frac{1}{e}$ and

$$\mathrm{E}\left(g(X_{t+1}) - g(X_t) \mid X_t, \lambda_t \le en\right) \ge \frac{1}{e} - \frac{s}{e-1}.$$

For the case $\lambda_t > en$, in an unsuccessful generation the penalty term is capped at its maximum and we pessimistically bound the positive effect on the potential from below by 0. However, the probability of a fitness improvement is large enough to show a positive drift: by Lemma 2.2, $\lambda_t > en$ implies $p^+_{i,\lambda} \geq 1 - \left(1 - \frac{1}{en}\right)^{en} \geq 1 - \frac{1}{e}$ and $p^-_{i,\lambda}\Delta^-_{i,\lambda} \leq \left(\frac{e-1}{e}\right)^{en} \frac{e}{e-1} = \left(\frac{e-1}{e}\right)^{en-1}$. Together,

$$\begin{aligned}
&\mathrm{E}\left(g(X_{t+1}) - g(X_t) \mid X_t, \lambda_t > en\right)\\
&\geq p^+_{i,\lambda}\left(\Delta^+_{i,\lambda} - \frac{se}{e-1}\right) + p^-_{i,\lambda}\left(-\Delta^-_{i,\lambda}\right)\\
&\geq \left(1 - \frac{1}{e}\right)\left(1 - \frac{se}{e-1}\right) - \left(\frac{e-1}{e}\right)^{en-1}.
\end{aligned}$$

Since $\left(1 - \frac{1}{e}\right) = \frac{1}{e} + \left(1 - \frac{2}{e}\right)$ and $\left(1 - \frac{2}{e}\right)\left(1 - \frac{se}{e-1}\right)$ is a positive constant, for large enough n this is larger than $\left(\frac{e-1}{e}\right)^{en-1}$ and

$$\mathrm{E}\left(g(X_{t+1}) - g(X_t) \mid X_t, \lambda_t > en\right) \geq \frac{1}{e}\left(1 - \frac{se}{e-1}\right) = \frac{1}{e} - \frac{s}{e-1}.$$

Since $s < \frac{e-1}{e}$, this is a strictly positive constant. □

With this constant lower bound on the drift of the potential, the proof of Theorem 3.1 is now quite straightforward.

Proof of Theorem 3.1. We bound the time to get to optimum using the potential function $g(X_t)$. Lemma 3.4 shows that the potential has a positive constant drift whenever the optimum has not been found, and by Lemma 3.3 if $g(X_t) = n$ then the optimum has been found. Therefore, we can bound the number of generations by the time it takes for $g(X_t)$ to reach n.

To fit the perspective of the additive drift theorem [13] we switch to the function $\overline{g}(X_t) := n - g(X_t)$ and note that $\overline{g}(X_t) = 0$ implies that $g(X_t) = f(x_t) = n$. The initial value $\overline{g}(X_0)$ is at most $n + \frac{se}{e-1}\log_F\left(enF^{1/s}\right)$ by Lemma 3.3. Using Lemma 3.4 and the additive drift theorem, the expected number of generations is at most

$$\frac{n + \frac{se}{e-1}\log_F\left(enF^{1/s}\right)}{\frac{1}{e} - \frac{s}{e-1}} = O(n). \quad □$$

3.2 Bounding the Number of Evaluations

A bound on the number of generations, by itself, is not sufficient to claim that the self-adjusting $(1,\lambda)$ EA is efficient. Since λ grows exponentially in unsuccessful generations, it could quickly attain very large values. However, we show that this is not the case and only $O(n \log n)$ evaluations are sufficient, in expectation.

Theorem 3.5. *Let the update strength $F > 1$ and the success rate $0 < s < \frac{e-1}{e}$ be constants. The expected number of function evaluations of the self-adjusting $(1,\lambda)$ EA on OneMax is $O(n \log n)$.*

Bounding the number of evaluations is more challenging than bounding the number of generations as we need to keep track of the offspring population size λ and how it develops over time. Large values of λ lead to a large number of evaluations made in one generation. Small values of λ can lead to a fallback.

If our algorithm was elitist, small values of λ would not be an issue since there would be no fallbacks. We will show later on (Lemma 3.10) that then the algorithm will spend $O(n \log n)$ evaluations, refining the amortised analysis from Lässig and Sudholt [18]. This analysis relies on every fitness level being visited at most once. In our non-elitist algorithm, this is not guaranteed. Small values of λ can lead to decreases in fitness, and then the same fitness level can be visited multiple times.

The reader may think that small values of λ only incur few evaluations and that the additional cost for a fallback is easily accounted for. However, it is not that simple. Imagine a fitness level i and a large value of λ such that a fallback is unlikely. But it is possible for λ to decrease in a sequence of improving steps. Then we would have a small value of λ and possibly a sequence of fitness-decreasing steps. Suppose the fitness decreases to a value at most i, then if λ returns to a large value, we may have visited fitness level i multiple times, with large (and costly) values of λ.

It is possible to show that, for sufficiently challenging fitness levels, λ moves towards an equilibrium state, i. e. when λ is too small, it tends to increase. However, this is generally not enough to exclude drops in λ. Since λ is multiplied or divided by a constant factor in each step, a sequence of k improving steps decreases λ by a factor of F^k, which is exponential in k. For instance, a value of $\lambda = \log^{O(1)} n$ can decrease to $\lambda = \Theta(1)$ in only $O(\log \log n)$ generations. We found that standard techniques such as the negative drift theorem, applied to $\log_F(\lambda_t)$, are not strong enough to exclude drops in λ.

We solve this problem as follows. We consider the best-so-far fitness $f^*_t = \max\{f(x_{t'}) \mid 0 \leq t' \leq t\}$ at time t (as a theoretical concept, as the self-adjusting $(1,\lambda)$ EA is non-elitist and unaware of the best-so-far fitness) and use drift arguments from Section 3.1, and the negative drift theorem [26, 27] to show that, with high probability, the current fitness never drops far below f^*_t, that is, $f(x_t) \geq f^*_t - r\log n$ for a constant $r > 0$. This yields a *ratchet argument*: if the best-so-far fitness increases, the lower bound on the current fitness increases as well.

It remains to show that the best-so-far fitness increases efficiently. We divide the run into fitness intervals of size $\log n$ that we call *blocks*, and bound the time for the best-so-far fitness to reach a better block. This task is easier than bounding the time to go all the way to the optimum since it is sufficient to have a sequence of generations in which λ maintains large enough values (i. e. $\lambda \geq 4\log n$) such that with high probability the fitness does not decrease and the self-adjusting $(1,\lambda)$ EA temporarily behaves like an elitist algorithm. We show that, starting from an arbitrary λ-value, the probability of having such a sequence is $\Omega(1)$ and that the expected time to reach the next block is only by at most a constant factor larger than that of an elitist algorithm. Adding up expected times for each block then yields the claimed $O(n \log n)$ bound.

We first re-use the potential drift arguments from the proof of Theorem 3.1 to show that the number of *generations* to increase the current fitness to a new block is bounded as follows. For $b = a + \log n$, this bound is $O(\log n)$.

Lemma 3.6. *Consider the self-adjusting $(1,\lambda)$ EA as in Theorem 3.5. For every $a, b \in \mathbb{N}$, the expected number of generations to increase the current fitness from a value at least a to at least b is at most*

$$\frac{b - a + \frac{se}{e-1}\log_F\left(enF^{1/s}\right)}{\frac{1}{e} - \frac{s}{e-1}} = O(b - a + \log n).$$

Proof. We use the proof of Theorem 3.1 with a revised potential function of $\overline{g}'(X_t) := \max(\overline{g}(X_t) - (n-b), 0)$ and stopping when $\overline{g}'(X_t) = 0$ (which implies that a fitness of at least b is reached) or a fitness of at least b is reached beforehand. Note that the maximum caps the effect of fitness improvements that jump to fitness values larger than b. As remarked in Lemma 3.4, the drift bound for $g(X_t)$ still holds when only considering fitness improvements by 1. Hence, it also holds for $g'(X_t)$ and the analysis goes through as before. □

Now we show our ratchet argument and that with high probability the fitness does not decrease when $\lambda \geq 4 \log n$.

Lemma 3.7. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 3.5. Let $f_t^* := \max_{t' \leq t} f(x_{t'})$ be its best-so-far fitness at generation t and let T be the first generation in which the optimum is found. Then with probability $1 - O(1/n)$ the following statements hold for a large enough constant $r > 0$ (that may depend on s).*

(1) *For all $t \leq T$ in which $\lambda_t \geq 4 \log n$, we have $f(x_{t+1}) \geq f(x_t)$.*
(2) *For all $t \leq T$, the fitness is at least: $f(x_t) \geq f_t^* - r \log n$.*

Proof. Let E_1^t denote the event that $\lambda_t < 4 \log n$ or $f(x_{t+1}) \geq f(x_t)$. Hence we only need to consider λ_t-values of $\lambda_t \geq 4 \log n \geq 2 \log_{\frac{e}{e-1}} n$ and by Lemma 2.2 we have

$$\Pr\left(\overline{E_1^t}\right) \leq \left(\frac{e-1}{e}\right)^{\lambda_t} \leq \left(\frac{e-1}{e}\right)^{2 \log_{\frac{e}{e-1}} n} = \frac{1}{n^2}.$$

By a union bound, the probability that this happens in the first T generations is at most $\sum_{t=1}^{\infty} \Pr(T = t) \cdot t/n^2 = \mathrm{E}(T)/n^2 = O(1/n)$.

For the second statement, let t^* be a generation in which the best-so-far fitness was attained: $f(x_{t^*}) = f_t^*$. By Lemma 3.3, abbreviating $\alpha := \frac{se}{e-1} \log_F(enF^{1/s})$, the condition $f(x_{t^*}) \geq f(x_t) + r \log n$ implies $g(X_{t^*}) \geq f(x_{t^*}) - \alpha \geq f(x_t) - \alpha + r \log n \geq g(X_t) - \alpha + r \log n$.

Now define events $E_2^t = (\forall t' \in [t+1, n^2] \colon g(X_{t'}) \geq g(X_t) + \alpha - r \log(n))$. We apply the negative drift theorem [26, 27] to bound $\Pr\left(\overline{E_2^t}\right)$ from above. For any $t < n^2$ let $a := g(X_t) - r \log n + \alpha$ and $b := g(x_t) < n$, where $r > \alpha$ will be chosen later on. We pessimistically assume that the fitness component of g can only increase by at most 1. Lemma 3.4 has already shown that, even under this assumption, the drift is at least a positive constant. This implies the first condition of Theorem 2 in [27]. For the second condition, we need to bound transition probabilities for the potential. Owing to our pessimistic assumption, the current fitness can only increase by at most 1. The fitness only decreases by j if *all* offspring are worse than their parent by at least j. Hence, for all λ, the decrease in fitness is bounded by the decrease in fitness of the *first* offspring. The probability of the first offspring decreasing fitness by at least j is bounded by the probability that j bits flip, which is in turn bounded by $1/(j!) \leq 2/2^j$. The possible penalty in the definition of g changes by at most $\max\left(\frac{se}{e-1}, \frac{se}{e-1} \cdot \frac{1}{s}\right) = \frac{e}{e-1} < 1$. Hence, for all t,

$$\Pr\left(|g(X_{t-1}) - g(X_t)| \geq j + 1 \mid g(X_t) > a\right) \leq \frac{4}{2^{j+1}},$$

which meets the second condition of Theorem 2 in [27]. It then states that there is a constant c^* such that the probability that within $2^{c^*(a-b)/4}$ generations a potential of at most a is reached, starting from a value of at least b, is $2^{-\Omega(a-b)}$. By choosing the constant r large enough, we can scale up $a - b$ and thus make $2^{c^*(a-b)/4} \geq n^2$ and $2^{-\Omega(a-b)} = O(1/n^2)$. This yields $\Pr\left(\overline{E_2^t}\right) = O(1/n^2)$.

Arguing as before, the probability that $\overline{E_2^t}$ happens during $O(n)$ expected generations is $O(1/n)$. By Markov's inequality, the probability of not finding the optimum in n^2 generations is $O(1/n)$ as well. Adding up all failure probabilities completes the proof. □

The following lemma bounds the probability of increasing a small λ value to a desired one from below.

Lemma 3.8. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 3.5 and assume that the typical events stated in Lemma 3.7 occur. Let $i := f(x_t)$ and r be the constant from Lemma 3.7. Then for all $\lambda_{\text{new}} \geq \lambda_{\text{init}} \geq 1$ the probability that λ is increased from λ_{init} to λ_{new} in a sequence of $s \log_F(\lambda_{\text{new}}/\lambda_{\text{init}})$ non-improving generations is at least*

$$1 - \frac{\lambda_{\text{new}} p^+_{i-r\log n,1}}{F^{1/s} - 1}$$

Proof. While no improvement is found, λ is multiplied by $F^{1/s}$ in every iteration. Then λ reaches a value of λ_{new} from λ_{init} in $\gamma := s \log_F(\lambda_{\text{new}}/\lambda_{\text{init}})$ iterations (since $\lambda_{\text{init}} \cdot F^{s \log_F(\lambda_{\text{new}}/\lambda_{\text{init}})/s} = \lambda_{\text{new}}$). The number of evaluations made during this time is at most

$$\begin{aligned}\sum_{j=0}^{\gamma-1} \lambda_{\text{init}} \cdot F^{j/s} = \lambda_{\text{init}} \cdot \sum_{j=0}^{\gamma-1} (F^{1/s})^j &= \lambda_{\text{init}} \cdot \frac{(F^{1/s})^\gamma - 1}{F^{1/s} - 1} \\ &= \lambda_{\text{init}} \cdot \frac{\lambda_{\text{new}}/\lambda_{\text{init}} - 1}{F^{1/s} - 1} \leq \frac{\lambda_{\text{new}}}{F^{1/s} - 1}.\end{aligned}$$

Since, owing to Lemma 3.7, the fitness is always at least $i - r \log n$ and $p^+_{i,1}$ is non-increasing in i, the probability of an improvement during any offspring creation is at most $p^+_{i-r\log n,1}$. By a union bound, the probability of having an improvement during $\frac{\lambda_{\text{new}}}{F^{1/s}-1}$ mutations of a search point with fitness i is at most $\frac{\lambda_{\text{new}} p^+_{i-r\log n,1}}{F^{1/s}-1}$. □

Now we bound the probability of returning to a small value of λ.

Lemma 3.9. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 3.5 and assume that the typical events stated in Lemma 3.7 occur. If the current fitness is $f(x_t) \geq n - n/\log^3 n$ and $\lambda_t \geq 4 \log^3 n$, then, for every constant $C > 0$, the probability that within $C \log n$ iterations a λ-value of at most $4 \log n$ is reached is at most $(\log n)^{-\omega(1)}$.*

Proof. We assume that $\lambda_t < 4F \log^3 n$ as otherwise we can wait until λ drops below $4F \log^3 n$ (or $C \log n$ generations have passed).

By Lemma 3.7, while $\lambda \geq 4 \log n$ the current fitness does not decrease. Consequently, the probability of an offspring finding an improvement is always bounded from above by $p^+_{i,1} \leq \frac{1}{\log^3 n}$.

For $k := 2 \log_F(\log n)$ we have $4 \log^3(n)/F^k = 4 \log n$. A necessary condition to decrease λ from a value at least $4 \log^3 n$ to a value below $4 \log n$ is that for every $j \in \{0, \dots, k-1\}$ there exist generations in which an improvement is found while $\lambda \leq 4 \log^3(n)/F^j$. Then, by a union bound, the probability of finding an improvement is at most $4 \log^3(n)/F^j \cdot p^+_{i,1}$. There are $\binom{C \log n}{k}$ ways to choose these generations. Once these are fixed, the joint probability of having improvements in all these iterations is at most

$$\prod_{j=0}^{k-1}\left(\frac{4F\log^3 n}{F^j}\cdot p_{i,1}^+\right) \le \left(4F\log^3(n)p_{i,1}^+\right)^k F^{-\sum_{j=0}^{k-1} j} \le (4F)^k F^{-k(k-1)/2}.$$

Along with a factor of $\binom{C\log n}{k} \le (eC\log(n)/k)^k$ we bound the sought probability as

$$\left(\frac{\lambda_t p_{i,1}^+ eC\log n}{kF^{(k-1)/2}}\right)^k \le \left(\frac{4F\log^3(n)\cdot 1/(\log^3 n)\cdot eC\log n}{kF^{-1/2}\log n}\right)^k$$
$$= \left(\frac{4eCF^{3/2}}{k}\right)^k = (\log\log n)^{-\Omega(\log\log n)} = (\log n)^{-\Omega(\log\log\log n)}. \qquad \square$$

While the fitness does not decrease, the self-adjusting $(1,\lambda)$ EA behaves like an elitist algorithm, i. e. a $(1+\{F^{1/s}\lambda, \lambda/F\})$ EA. We bound its expected time to increase the fitness from a to b.

Lemma 3.10. *Consider the elitist $(1+\{F^{1/s}\lambda, \lambda/F\})$ EA on OneMax with $f(x_0) \ge a$ and $\lambda_0 = \lambda_{\text{init}}$. For every $b \in \mathbb{N}$, the expected number of evaluations for it to reach a fitness of at least b is at most*

$$\lambda_{\text{init}}\cdot\frac{F}{1-F} + n\cdot\frac{(2+2e)F^{1/s}-1}{F^{1/s}-1}\cdot\frac{F^{\frac{s+1}{s}}-1}{F-1}\sum_{j=a}^{b-1}\frac{1}{n-j}.$$

As a side note, the lemma immediately implies the following.

Corollary 3.11. *The expected number of evaluations of the elitist $(1+\{F^{1/s}\lambda, \lambda/F\})$ EA on OneMax is $O(n\log n)$.*

Proof of Lemma 3.10. We refine the *accounting method* used in the analysis of the $(1+\{2\lambda, \lambda/2\})$ EA in [18]. The main idea is: if some fitness level i increases λ to a large value, we charge the costs for increasing λ to that fitness level. In addition, we charge costs that pay for decreasing λ down to 1 in future successful generations. Hence, a successful generation comes for free, at the expense of a constant factor for the cost of unsuccessful generations.

Let $\phi(\lambda_t) := \frac{F}{F-1}\lambda_t$ and imagine a fictional bank account. We make an initial payment of $\phi(\lambda_{\text{init}})$ to that bank account. If a generation t is unsuccessful, we pay λ_t for the current generation and deposit an additional amount of $\phi(\lambda_t F^{1/s}) - \phi(\lambda_t)$. If an improvement is found, we withdraw an amount of λ_t to pay for generation t.

We show by induction: for every generation t, the account's balance is $\phi(\lambda_t)$. This is true for the initial generation owing to the initial payment of $\phi(\lambda_{\text{init}})$. Assume the statement holds for time t. If generation t is unsuccessful, the new balance is

$$\phi(\lambda_t) + (\phi(\lambda_t F^{1/s}) - \phi(\lambda_t)) = \phi(\lambda_t F^{1/s}) = \phi(\lambda_{t+1}).$$

If generation t is successful, $\lambda_{t+1} = \lambda_t/F$ and the new balance is

$$\phi(\lambda_t) - \lambda_t = \left(\frac{F}{F-1}-1\right)\lambda_t = \frac{1}{F-1}\cdot\lambda_t = \frac{F}{F-1}\cdot\lambda_{t+1} = \phi(\lambda_{t+1}).$$

The costs of an unsuccessful generation t are

$$\lambda_t + \phi(\lambda_t F^{1/s}) - \phi(\lambda_t) = \lambda_t\left(1+\frac{F^{\frac{s+1}{s}}}{F-1} - \frac{F}{F-1}\right) = \lambda_t\cdot\frac{F^{\frac{s+1}{s}}-1}{F-1}.$$

We show that the expected number of evaluations *in unsuccessful generations* on fitness level i is at most $\frac{n}{n-i}\cdot\frac{(2+2e)F^{1/s}-1}{F^{1/s}-1}$, for all initial λ. Multiplying with the above factor of $\frac{F^{\frac{s+1}{s}}-1}{F-1}$ and adding up costs for all fitness levels as well as costs $\phi(\lambda_{\text{init}}) = \lambda_{\text{init}}\cdot\frac{F}{1-F}$ for the initial λ yields the claimed bound.

After j generations the offspring population size is $\lambda F^{j/s}$. We only count these terms if there has been no success. The expected number of evaluations in unsuccessful generations is at most

$$\sum_{j=0}^{\infty}\lambda F^{j/s}\cdot\Pr\left(\text{no success with }\lambda F^{j/s}\text{ offspring}\right)$$
$$\le \sum_{j=0}^{\infty}\lambda F^{j/s}\cdot\left(1-\frac{n-i}{en}\right)^{\lambda F^{j/s}} \le \sum_{j=0}^{\infty}\lambda F^{j/s}\cdot e^{-\frac{n-i}{en}\cdot\lambda F^{j/s}}. \qquad (5)$$

Let $k \in \mathbb{N}_0$ be the smallest integer such that $\frac{n-i}{en}\cdot\lambda F^{k/s} \ge 2$. The function $\xi(x) := x\cdot e^{-px}$ attains its maximum $1/(ep)$ at $x = 1/p$ and for $px \ge 2$ it decays exponentially:

$$\frac{\xi(x)}{\xi(xF^{1/s})} = \frac{e^{(F^{1/s}-1)px}}{F^{1/s}} \ge \frac{e^{2F^{1/s}-2}}{F^{1/s}} \ge \frac{2F^{1/s}-1}{F^{1/s}} = 2 - F^{-1/s} > 1.$$

Bounding the summand $j = k$ in (5) by ξ's maximum, $\frac{1}{e}\cdot\frac{en}{n-i} = \frac{n}{n-i}$, and exploiting the exponential decay for all following terms,

$$\sum_{j=k}^{\infty}\lambda F^{j/s}\cdot e^{-\frac{n-i}{en}\cdot\lambda F^{j/s}} \le \sum_{j=k}^{\infty}\frac{n}{n-i}\cdot(2-F^{-1/s})^{k-j}$$
$$= \frac{n}{n-i}\cdot\frac{2-F^{-1/s}}{1-F^{-1/s}} = \frac{n}{n-i}\cdot\frac{2F^{1/s}-1}{F^{1/s}-1}.$$

For $k > 0$ we bound the terms for $j < k$, using $\frac{n-i}{en}\cdot\lambda F^{(k-1)/s} < 2$ by definition of k,

$$\sum_{j=0}^{k-1}\lambda F^{j/s}\cdot e^{-\frac{n-i}{en}\cdot\lambda F^{j/s}} \le \sum_{j=0}^{k-1}\lambda F^{j/s} = \sum_{j=0}^{k-1}\lambda F^{(k-1)/s}\cdot F^{-j/s}$$
$$\le \sum_{j=0}^{k-1}\frac{2en}{n-i}\cdot F^{-j/s} \le \frac{2en}{n-i}\sum_{j=0}^{\infty}F^{-j/s} = \frac{2en}{n-i}\cdot\frac{F^{1/s}}{F^{1/s}-1}.$$

The sum in (5) is at most $\frac{n}{n-i}\cdot\frac{(2+2e)F^{1/s}-1}{F^{1/s}-1}$, completing the proof. $\square$

Lemma 3.12. *Consider the self-adjusting $(1,\lambda)$ EA as in Theorem 3.5 and assume that the typical events stated in Lemma 3.7 occur. Let f_t^* be the best-so-far fitness at time t. If $f_t^* \ge n - n/\log^3 n$ and $f_t^* \le n - \log n$, the expected number of evaluations made until the best-so-far fitness increases to at least $f_t^* + \log n$ is at most*

$$\beta\cdot\left(\lambda_t + \log^3(n)\log\log(n) + \sum_{j=f_t^*-r\log n}^{f_t^*+\log n}\frac{n}{n-j}\right)$$

for some constant $\beta > 0$ (that depends on s and F).

Proof. We first show that the expected number of evaluations until either a fitness of at least $f_t^* + \log n$ is reached or $\lambda \ge 4\log^3 n$ is reached is bounded by $O(\log^3(n)\log\log n)$.

By Lemma 3.8, the probability that in a sequence of $O(\log\log n)$ generations a λ-value of at least $4\log^3 n$ is reached is at least

$$1 - \frac{4\log^3(n)p_{i-r\log n,1}^+}{F^{1/s}-1} \ge 1 - \frac{4\log^3(n)}{F^{1/s}-1}\cdot\frac{n/\log^3 n + r\log n}{en} = \Omega(1).$$

Hence, the expected number of phases of $O(\log\log n)$ generations for this to happen is $O(1)$. As every generation makes at most $4F^{1/s}\log^3 n$ evaluations, the expected number of evaluations is $O(\log^3(n)\log\log n)$.

Now assume $\lambda \geq 4\log^3 n$. We say that a failure occurs if λ drops below $4\log n$ before the fitness is increased to at least $f_t^* + \log n$. By Lemma 3.6, the expected number of generations to increase the fitness from f_t^* to at least $f_t^* + \log n$ is at most $c_{\text{gen}}\log n$, for a constant c_{gen}. By Markov's inequality, the probability that more than $2c_{\text{gen}}\log n$ generations are needed is at most $1/2$. By Lemma 3.9, the probability of decreasing λ to less than $4\log n$ in the next $2c_{\text{gen}}\log n$ generations is $(\log n)^{-\omega(1)}$. By a union bound, the probability of a failure is at most $1/2 + (\log n)^{-\omega(1)}$.

While no failure occurs, the algorithm behaves like a $(1 + \{F^{1/s}\lambda, \lambda/F\})$ EA. Applying Lemma 3.10 with $a := f_t^* - r\log n$ and $b := f_t^* + \log n$, the expected number of evaluations until a fitness of at least $f_t^* + \log n$ is found or a fallback occurs is at most

$$\lambda_{\text{init}} \cdot \frac{F}{1-F} + n \cdot \frac{(2+2e)F^{1/s} - 1}{F^{1/s} - 1} \cdot \frac{F^{\frac{s+1}{s}} - 1}{F - 1} \sum_{j=f_t^* - r\log n}^{f_t^* + \log n - 1} \frac{1}{n-j}.$$

In case of a failure we iterate the above arguments; this increases the expected number of evaluations only by a factor $2 + o(1)$. □

Lemma 3.13. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 3.5. If the best-so-far fitness at time t is at most i then*

$$E(\lambda_t \mid \lambda_0) \leq \lfloor \lambda_0/F^t \rfloor + \frac{en}{n-i} \cdot \left(F^{1/s} + \frac{F^{1/s}}{\ln F}\right).$$

Proof sketch for Lemma 3.13. Since the fitness in all generations $t' \leq t$ is at most i, the probability of one offspring finding an improvement at time t' is at least $\frac{n-i}{en}$. We have $\mathrm{E}(\lambda_t \mid \lambda_0) = \sum_{x=1}^{\infty} \Pr(\lambda_t \geq x \mid \lambda_0)$ and bound the latter probabilities from above. If $\lambda_0 \geq x \cdot F^t$ then $\lambda_t \geq x$ with probability 1. If $\lambda_0 < x \cdot F^t$, $\lambda_t \geq x$ only holds if there has been at least one generation where λ was increased. Let $t - k > 0$ be the last such generation, that is, $\lambda_{t-k} = \lambda_{t-k-1}F^{1/s}$. Since all generations $t-k, \dots, t-1$ are successful, $\lambda_t \geq x$ implies $\lambda_{t-k} \geq xF^k$ and hence $\lambda_{t-k-1} \geq xF^{k-1/s}$.

For all $x > \lambda_0/F^t$, $\Pr(\lambda_t \geq x \mid \lambda_0)$ is at most

$$\sum_{k=1}^{\infty} \Pr\left(\text{no success with } \frac{xF^k}{F^{1/s}} \text{ offspring}\right) \leq \sum_{k=1}^{\infty} \left(1 - \frac{n-i}{en}\right)^{\frac{xF^k}{F^{1/s}}}.$$

Bounding this by $\left(1 - \frac{n-i}{en}\right)^{x/F^{1/s}}$ for $x > \lambda_0/F^t$ and $x \geq \frac{en}{n-i} \cdot \frac{F^{1/s}}{\ln F}$ and using $\Pr(\lambda_t \geq x \mid \lambda_0) \leq 1$ for $x \leq \max\left(\lfloor \lambda_0/F^t \rfloor, \left\lfloor \frac{en}{n-i} \cdot \frac{F^{1/s}}{\ln F} \right\rfloor\right)$ proves the claim. □

Now we are in a position to prove Theorem 3.5.

Proof of Theorem 3.5. We divide the optimisation in several phases. Phase 1 ends when the distance to the optimum is at most $n/\log n$. By Lemma 3.6 the expected number of *generations* spent in Phase 1, called T_1, is $\mathrm{E}(T_1) = O(n)$.

The expected number of function evaluations during this time is $\mathrm{E}(\lambda_0 + \lambda_1 + \dots + \lambda_{T_1 - 1}) = \sum_{t=0}^{T_1-1} \mathrm{E}(\lambda_t \mid \lambda_0)$. We bound all summands by Lemma 3.13, applied with a worst case fitness of $i := n - n/\log n$. This yields a random variable λ^* with

$$\mathrm{E}(\lambda^*) \leq \frac{en}{n/\log n} \cdot \left(F^{1/s} + \frac{F^{1/s}}{\ln F}\right) = e\log n \cdot \left(F^{1/s} + \frac{F^{1/s}}{\ln F}\right)$$

and $\mathrm{E}(\lambda^*) \geq \mathrm{E}(\lambda_t \mid \lambda_0)$ for all $t < T_1$. Thus, the expected time in Phase 1 can be bounded by T_1 iid variables λ^*. Since T_1 is itself a random variable, we apply Wald's equation [33] to conclude that

$$\sum_{t=0}^{T_1-1} \mathrm{E}(\lambda^*) = \mathrm{E}(T_1) \cdot \mathrm{E}(\lambda^*) = O(n\log n).$$

Phase 2 ends when the distance to the optimum is at most $n/\log^2 n$. Again, by Lemma 3.6 the expected number of *generations* is T_2 with $\mathrm{E}(T_2) = O(n/\log n)$ and the expected number of evaluations in one generation is bounded by $e\log^2 n \cdot \left(F^{1/s} + \frac{F^{1/s}}{\ln F}\right)$ using Lemma 3.13 with a worst-case fitness of $i := n - n/\log^2 n$. Wald's equation then yields a bound of $O(n/\log n) \cdot O(\log^2 n) = O(n\log n)$.

Phase 3 ends with the distance to the optimum is at most $n/\log^3 n$ and we obtain another bound of $O(n\log n)$ in the same way.

Phase 4 ends when the optimum is found. We divide the distance to the optimum in *blocks* of length $\log n$. Let T_i be the random number of evaluations to increase the best fitness from at least $n - n/\log^3(n) + (i-1)\log n$ to a fitness of at least $n - n/\log^3(n) + i\log n$. Let λ_i^* be the λ-value in the first generation where this block i is reached. Then the expected number of evaluations to find the optimum from a best fitness of at least $n - n/\log^3 n$ is at most

$$\mathrm{E}\left(\sum_{i=1}^{n/\log^4 n} T_i\right) = \sum_{i=1}^{n/\log^4 n} \mathrm{E}(T_i) = \sum_{i=1}^{n/\log^4 n} \mathrm{E}\left(\mathrm{E}(T_i \mid \lambda_i^*)\right)$$

By Lemma 3.12, this is at most

$$\sum_{i=1}^{n/\log^4 n} \mathrm{E}\left(\beta\left(\lambda_i^* + \log^3(n)\log\log(n) + \sum_{j=n-n/\log^3(n)+(i-1-r)\log n}^{n-n/\log^3(n)+i\log(n)-1} \frac{1}{n-j}\right)\right).$$

The terms $\beta\log^3(n)\log\log n$ sum up to $O(n\log\log n)$. Note that

$$\beta \sum_{i=1}^{n/\log^4 n} \sum_{j=n-n/\log^3(n)+(i-1-r)\log n}^{n-n/\log^3(n)+i\log(n)-1} \frac{1}{n-j} \leq \beta \sum_{i=0}^{n-1} \frac{r+1}{n-j} = O(n\log n)$$

since every summand $\frac{1}{n-j}$ appears in at most $r+1$ blocks. Finally, by Lemma 3.13, $\sum_{i=1}^{n/\log^4 n} \mathrm{E}(\lambda_i^*) = O(n\log n)$ with room to spare.

In case a failure occurs, we repeat the above arguments. Note that then we start the analysis with λ_0 being the λ-value at the time of the failure. A large λ-value may incur additional costs. However, applying Lemma 3.13 with a fitness of $i = n$ shows that then the expected λ-value is at most $O(n)$. And, since the $\lfloor \lambda_0/F^t \rfloor$ term in Lemma 3.13 decays exponentially in t, a value $\lambda_0 = O(n)$ only incurs additional costs of at most $\sum_{t=0}^{\infty} \lambda_0/F^t = \lambda_0 \cdot \frac{F}{F-1}$ in subsequent generations. Since failures have a probability of $O(1/n)$, the expected number of repetitions is $1 + O(1/n)$ and all additional costs can be absorbed in the $O(n\log n)$ bound. □

4 LARGE SUCCESS RATES FAIL

We show that the choice of the success rate is crucial as when s is a large constant, the runtime becomes exponential.

Theorem 4.1. *Let the update strength $1.052 \leq F \leq 1.45$ and the success rate $s \geq 22$ be constants. With probability $1 - e^{-\Omega(n/\log^4 n)}$ the self-adjusting $(1, \lambda)$ EA needs at least $e^{\Omega(n/\log^4 n)}$ evaluations to optimise OneMax.*

The reason why the algorithm takes exponential time is that now $F^{1/s}$ is small and λ only increases slowly in unsuccessful generations, whereas successful generations decrease λ by a much larger factor of F. This is detrimental during early parts of the run where it is *easy* to find improvements and there are frequent improvements that decrease λ. When λ is small, there are frequent fallbacks, hence the algorithm stays in a region with small values of λ, where it finds improvements with constant probability, but also has fallbacks with constant probability. We show, using another potential function, that it takes exponential time to escape from this equilibrium.

Definition 4.2. We define the potential function $h(X_t)$ as

$$h(X_t) := f(x_t) + 2.1 \log_F^2 \lambda_t.$$

While $g(X_t)$ used a linear contribution of $\log_F(\lambda_t)$, here we use a convex function of $\log_F^2(\lambda_t)$, so that decreases of λ_t have a larger impact on the potential. We show that, in a given fitness interval, the potential $h(X_t)$ has a negative drift.

Lemma 4.3. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 4.1. Then there is a constant $\delta > 0$ that for every $0.9n + 2.1 \log^2(4.5) < h(X_t) < 0.91n$,*

$$E\left(h(X_{t+1}) - h(X_t) \mid X_t\right) \leq -\delta.$$

Proof sketch for Lemma 4.3. We abbreviate $\Delta_h := E\left(h(X_{t+1}) - h(X_t) \mid X_t\right)$ and bound it by considering the possible outcomes of a generation. The differences in the λ-terms of $h(X_t)$ in successful and unsuccessful generations, respectively, are $2.1 \log_F^2(\lambda/F) - 2.1 \log_F^2(\lambda) = -4.2 \log_F(\lambda) + 2.1$ and $2.1 \log_F^2(\lambda F^{1/s}) - 2.1 \log_F^2(\lambda) = \frac{4.2 \log_F(\lambda)}{s} + \frac{2.1}{s}$. We show that

$$\Delta_h \leq \left(\Delta^+_{i,\lambda} - 4.2 \log_F(\lambda) + 2.1 - \frac{4.2 \log_F \lambda}{s} - \frac{2.1}{s^2}\right) p^+_{i,\lambda} + \frac{4.2 \log_F \lambda}{s} + \frac{2.1}{s^2} - p^-_{i,\lambda}. \quad (6)$$

Then we consider different ranges of λ values and show that in every range the drift is a negative constant. Since we deal with small λ, we do account for rounding λ to its nearest integer in probability and drift bounds. Values of the drift computed using Equation 6 and the bounds from Lemma 2.2 are shown in Figure 1. For every value of λ in Figure 1, we used worst-case values for F and s to compute the bound on Δ_h. The maximum is -0.002. □

Finally, with Lemma 4.3, we now prove Theorem 4.1.

Proof sketch of Theorem 4.1. By Lemma 4.3 there is an interval of size $\Omega(n)$ for h for which the drift is a negative constant and by definition $f(x_t) \leq h(X_t)$. Therefore, to prove an exponential runtime we apply the negative drift theorem with scaling [28] to the potential function, showing that with the claimed probability

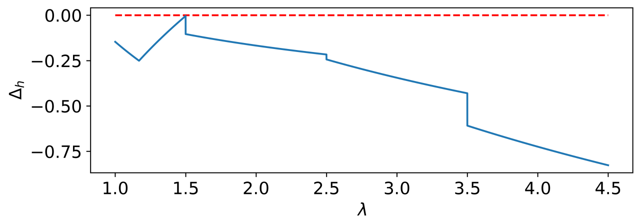

Figure 1: Bounds on Δ_h with a maximum of -0.002 for $\lambda = 1.5$.

we have $h(X_t) \leq 0.91n$ and thus $f(x_t) \leq 0.91n$. To meet the requirements of the negative drift theorem we bound the possible changes in the reward term by showing that with overwhelming probability $\lambda = O(n^3)$ while optimising OneMax and for the fitness we bound the tail of the number of flipping bits. Adding up changes in the fitness component and the reward component, we get $\Pr\left(|h(X_{t+1}) - h(X_t)| \mid X_t \geq j \cdot 20 \log^2 n\right) \leq e^{-j}$ for all $j \in \mathbb{N}$, meeting the requirements of the negative drift theorem. □

5 DISCUSSION AND CONCLUSIONS

We have shown that simple success-based rules, embedded in a $(1, \lambda)$ EA, are able to optimise OneMax in $O(n)$ generations and $O(n \log n)$ evaluations. The latter is best possible for any unary unbiased black-box algorithm [9, 19].

However, this result depends crucially on the correct selection of the success rate s. The above holds for constant $0 < s < \frac{e-1}{e}$ and, in sharp contrast, the time on OneMax becomes exponential with overwhelming probability if $s \geq 22$. Then the algorithm stagnates in an equilibrium state at a linear distance to the optimum. Simulations showed that, once λ grows large enough to escape from the equilibrium, the algorithm is able to maintain large values of λ until the optimum is found. Hence, we observe the counterintuitive effect that for too large values of s, optimisation is harder when the algorithm is far away from the optimum and becomes easier when approaching the optimum. (To our knowledge, such an effect was only reported before on HotTopic functions [21] and Dynamic BinVal functions [20].)

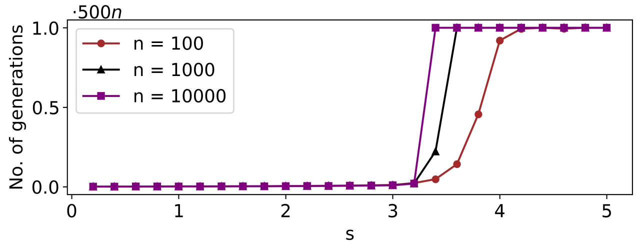

Figure 2: Average number of generations of the self-adjusting $(1, \lambda)$ EA with $F = 1.5$ in 100 runs for different n, normalised and capped at $500n$ generations.

There is a gap between conditions $s < \frac{e-1}{e}$ and $s \geq 22$. Further work is needed to close this gap. In preliminary experiments (Figure 2) we found a sharp threshold at $s \approx 3.4$, indicating that the widely used one-fifth rule ($s = 4$) is inefficient here, but other success rules achieve optimal asymptotic runtime.

REFERENCES

[1] Golnaz Badkobeh, Per Kristian Lehre, and Dirk Sudholt. 2014. Unbiased Black-Box Complexity of Parallel Search. In *Proc. of Parallel Problem Solving from Nature – PPSN XIII*. Springer, 892–901.

[2] 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 XI*, Vol. 6238. Springer, 1–10.

[3] B. Case and P. K. Lehre. 2020. Self-Adaptation in Nonelitist Evolutionary Algorithms on Discrete Problems with Unknown Structure. *IEEE Transactions on Evolutionary Computation* 24, 4 (2020), 650–663.

[4] Benjamin Doerr and Carola Doerr. 2018. Optimal Static and Self-Adjusting Parameter Choices for the (1+(λ,λ)) Genetic Algorithm. *Algorithmica* 80, 5 (2018), 1658–1709.

[5] Benjamin Doerr and Carola Doerr. 2020. *Theory of Parameter Control for Discrete Black-Box Optimization: Provable Performance Gains Through Dynamic Parameter Choices*. Springer, 271–321.

[6] Benjamin Doerr, Carola Doerr, and Franziska Ebel. 2015. From Black-Box Complexity to Designing New Genetic Algorithms. In *Theoretical Computer Science*, Vol. 567. 87–104.

[7] Benjamin Doerr, Carola Doerr, and Timo Kötzing. 2016. Provably Optimal Self-adjusting Step Sizes for Multi-valued Decision Variables. In *Proc. of Parallel Problem Solving from Nature – PPSN XIV*. Springer, 782–791.

[8] Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2019. Self-adjusting Mutation Rates with Provably Optimal Success Rules. In *Proceedings of the Genetic and Evolutionary Computation (GECCO ’19)*. ACM.

[9] Benjamin Doerr, Carola Doerr, and Jing Yang. 2020. Optimal Parameter Choices via Precise Black-Box Analysis. *Theoretical Computer Science* 801 (2020), 1 – 34.

[10] Benjamin Doerr, Christian Gießen, Carsten Witt, and Jing Yang. 2019. The (1+ λ) Evolutionary Algorithm with Self-Adjusting Mutation Rate. *Algorithmica* 81, 2 (2019), 593–631.

[11] 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 *Proceedings of the Genetic and Evolutionary Computation (GECCO ’18)*. ACM, 1015–1022.

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

[13] Jun He and Xin Yao. 2004. A Study of Drift Analysis for Estimating Computation Time of Evolutionary algorithms. *Natural Computing* 3, 1 (2004), 21–35.

[14] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2020. On the Choice of the Parameter Control Mechanism in the (1+(λ, λ)) Genetic Algorithm. In *Proceedings of the Genetic and Evolutionary Computation (GECCO ’20)*. ACM, 832–840.

[15] Mario Alejandro Hevia Fajardo and Dirk Sudholt. 2021. Self-Adjusting Population Sizes for Non-Elitist Evolutionary Algorithms: Why Success Rates Matter. *ArXiv e-prints* (2021). arXiv:2104.05624

[16] Thomas Jansen and Dirk Sudholt. 2010. Analysis of an Asymmetric Mutation Operator. *Evolutionary Computation* 18, 1 (2010), 1–26.

[17] 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, 1 (2004), 77–112.

[18] Jörg Lässig and Dirk Sudholt. 2011. Adaptive Population Models for Offspring Populations and Parallel Evolutionary Algorithms. In *Proceedings of the 11th Workshop Proceedings on Foundations of Genetic Algorithms (FOGA ’11)*. ACM, 181–192.

[19] Per Kristian Lehre and Carsten Witt. 2012. Black-Box Search by Unbiased Variation. *Algorithmica* 64, 4 (2012), 623–642.

[20] Johannes Lengler and Simone Riedi. 2021. Runtime Analysis of the (μ + 1)-EA on the Dynamic BinVal Function. In *Evolutionary Computation in Combinatorial Optimization*. Springer, 84–99.

[21] Johannes Lengler and Xun Zou. 2019. Exponential Slowdown for Larger Populations: The (μ + 1)-EA on Monotone Functions. In *Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms (FOGA ’19)*. ACM, 87–101.

[22] Andrei Lissovoi, Pietro Oliveto, and John Alasdair Warwicker. 2020. How the Duration of the Learning Period Affects the Performance of Random Gradient Selection Hyper-Heuristics. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 34. 2376–2383.

[23] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. 2019. On the Time Complexity of Algorithm Selection Hyper-Heuristics for Multimodal Optimisation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 2322–2329.

[24] Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. 2020. Simple Hyper-heuristics Control the Neighbourhood Size of Randomised Local Search Optimally for LeadingOnes. *Evolutionary Computation* 28, 3 (2020), 437–461.

[25] Andrea Mambrini and Dirk Sudholt. 2015. Design and Analysis of Schemes for Adapting Migration Intervals in Parallel Evolutionary Algorithms. *Evolutionary Computation* 23, 4 (2015), 559–582.

[26] Pietro S. Oliveto and Carsten Witt. 2011. Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *Algorithmica* 59, 3 (2011), 369–386.

[27] P. S. Oliveto and C. Witt. 2012. Erratum: Simplified Drift Analysis for Proving Lower Bounds in Evolutionary Computation. *ArXiv e-prints* (2012). arXiv:1211.7184

[28] Pietro S. Oliveto and Carsten Witt. 2015. Improved Time Complexity Analysis of the Simple Genetic Algorithm. *Theoretical Computer Science* 605 (2015), 21 – 41.

[29] Amirhossein Rajabi and Carsten Witt. 2020. Evolutionary Algorithms with Self-adjusting Asymmetric Mutation. In *Proc. of Parallel Problem Solving from Nature – PPSN XVI*. Springer, 664–677.

[30] Amirhossein Rajabi and Carsten Witt. 2020. Self-Adjusting Evolutionary Algorithms for Multimodal Optimization. In *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO ’20)*. ACM, 1314–1322.

[31] Amirhossein Rajabi and Carsten Witt. 2021. Stagnation Detection with Randomized Local Search. In *Evolutionary Computation in Combinatorial Optimization*. Springer, 152–168. Full version available at http://arxiv.org/abs/2101.12054.

[32] Jonathan E. Rowe and Dirk Sudholt. 2014. The Choice of the Offspring Population Size in the $(1, \lambda)$ Evolutionary Algorithm. *Theoretical Computer Science* 545 (2014), 20–38.

[33] Abraham Wald. 1944. On Cumulative Sums of Random Variables. *The Annals of Mathematical Statistics* 15, 3 (1944), 283 – 296.