Artificial Intelligence 328 (2024) 104061

Contents lists available at ScienceDirect

Artificial Intelligence

journal homepage: www.elsevier.com/locate/artint

Self-adjusting offspring population sizes outperform fixed parameters on the cliff function

Mario Alejandro Hevia Fajardo [a,*], Dirk Sudholt [b]

[a] *Department of Computer Science, University of Sheffield, Sheffield, United Kingdom*
[b] *Chair of Algorithms for Intelligent Systems, University of Passau, Passau, Germany*

ARTICLE INFO

Keywords:
Parameter control
Runtime analysis
Non-elitism
Drift analysis
Multimodal optimisation

ABSTRACT

In the discrete domain, self-adjusting parameters of evolutionary algorithms (EAs) have emerged as a fruitful research area with many runtime analyses showing that self-adjusting parameters can outperform the best fixed parameters. Most existing runtime analyses focus on elitist EAs on simple problems, for which moderate performance gains were shown. Here we consider a much more challenging scenario: the multimodal function Cliff, defined as an example where a $(1, \lambda)$ EA is effective, and for which the best known upper runtime bound for standard EAs is $O(n^{25})$.
We prove that a $(1, \lambda)$ EA self-adjusting the offspring population size λ using success-based rules optimises Cliff in $O(n)$ expected generations and $O(n \log n)$ expected evaluations. Along the way, we prove tight upper and lower bounds on the runtime for fixed λ (up to a logarithmic factor) and identify the runtime for the best fixed λ as n^{η} for $\eta \approx 3.97677$ (up to sub-polynomial factors). Hence, the self-adjusting $(1, \lambda)$ EA outperforms the best fixed parameter by a factor of at least $n^{2.9767}$.

1. Introduction

Evolutionary algorithms (EAs) as well as other Randomised Search Heuristics are used to solve a wide range of problems in part owing to their ease of implementation and their effectiveness in problems with little a priori knowledge. When applying an EA, a crucial task is to determine suitable parameters for the problem in hand such as mutation rate, crossover probability, population sizes, among others. In fact, it is well understood that the efficiency of the algorithms may depend drastically on their parameters [1], even to the point where small changes to the parameters can increase the runtime from polynomial to exponential [2,3]. To make matters worse, the optimal parameters may change during the optimisation problem, hence any static parameter choice may be sub-optimal [1].

Parameter control mechanisms aim to solve this problem by using dynamic parameter settings that adjust to the current state of the optimisation identifying and tracking the optimal parameter settings. Doerr and Doerr [1] have classified them into several types; here we focus on *success-based* parameter control mechanisms for their simplicity. In continuous optimisation, parameter control mechanisms have been used as standard for several decades because it is crucial to ensure convergence to the optimum. In contrast, in the discrete domain parameter control had not been as widely used in the past. In recent years it has become more common and developing a theory that explains and predicts the performance of parameter control mechanisms in the discrete domain

* Corresponding author.
E-mail address: maheviafajardo1@sheffield.ac.uk (M.A. Hevia Fajardo).

https://doi.org/10.1016/j.artint.2023.104061
Received 7 December 2022; Received in revised form 15 December 2023; Accepted 17 December 2023

Available online 21 December 2023

was highlighted as a major challenge and a hot topic in evolutionary computation by leading researchers in the field [4,5]. This is in part owing to runtime analyses showing that parameter control mechanisms can outperform the best static parameter settings, see the survey [1].

Here we highlight some examples relevant to this work where parameter control mechanisms have been proposed, along with proven performance guarantees. Böttcher et al. [6] considered the test function $\textsc{LeadingOnes}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_i$ that counts the number of consecutive ones at the start of the bit string x. They showed that fitness-dependent mutation rates can improve the performance of the $(1+1)$ EA on LeadingOnes by a constant factor. Badkobeh et al. [7] presented an adaptive strategy for the mutation rate in the $(1+\lambda)$ EA that, for all values of λ, leads to provably optimal performance on OneMax. Lässig and Sudholt [8] presented adaptive schemes for choosing the offspring population size in $(1+\lambda)$ EAs and the number of islands in an island model. Doerr et al. [9] proposed a fitness-dependent offspring population size of $\lambda = \sqrt{n/(n-f(x))}$ for the $(1+(\lambda,\lambda))$ GA showing that it optimises OneMax in $O(n)$ evaluations which is asymptotically faster than any static parameter choice and proposed a self-adjusting mechanism based on the one-fifth rule that tracked the optimal parameter in experiments. Later, Doerr and Doerr [10] proved that the self-adjusting mechanism in the $(1+(\lambda,\lambda))$ GA has the same asymptotic runtime on OneMax as the fitness-dependent mechanism. Hevia Fajardo and Sudholt [11] 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 et al. [12] 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 [7]. Mambrini and Sudholt [13] adapted the migration interval in island models and showed that adaptation can reduce the communication effort beyond the best possible fixed parameter. Doerr et al. [14] 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 et al. [15] proposed a Generalised Random Gradient Hyper-Heuristic that uses a learning period of τ steps that can learn to adapt the neighbourhood size of Randomised Local Search optimally during the run on LeadingOnes. They proved that it has the best possible runtime achievable by any algorithm that uses the same low level heuristics. 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 [16], OneMax and Ridge [17]. Rajabi and Witt [18] used a self-adjusting asymmetric mutation that can provide a constant-factor speed-up on OneMax over asymmetric mutations [19] and obtained the same asymptotic performance on the generalised OneMax function. Rajabi and Witt [20] proposed a stagnation detection mechanism that raises the mutation rate when the algorithm is likely to have encountered a local optimum. The mechanism can be added to any existing EA; when added to the $(1+1)$ EA, the SD–$(1+1)$ EA has the same asymptotic runtime on Jump as the optimal parameter setting. In a follow up study, Rajabi and Witt [21] added the stagnation detection mechanism to the RLS obtaining a constant factor speed-up from the SD–$(1+1)$ EA.

Most of the above analyses concern elitist EAs; there are very few studies of parameter control mechanisms for EAs using non-elitist selection. The first runtime analysis to show an asymptotic speedup for parameter control mechanisms in non-elitist EAs was presented by Dang and Lehre [22], showing that for a tailored multimodal function a self-adaptive EA is able to adjust the mutation rate, leading to exponential speedup over EAs with static mutation rates. Case and Lehre [23] showed a speedup for a self-adaptive (μ,λ) EA on the LeadingOnes problem with unknown solution length. Similarly Doerr et al. [24] proved that a self-adaptive mechanism for the mutation rate in the $(1,\lambda)$ EA with a sufficiently large λ has the same asymptotic expected runtime on OneMax as in [7]. Lissovoi et al. [25] proposed a hyper-heuristic that chooses between elitist and non-elitist heuristics that achieves the best known expected runtime for general purpose randomised search heuristics on the problem class $\textsc{Cliff}_d$. The present authors proved that a self-adjusting mechanism based on the one-fifth rule is able to find and maintain suitable parameter values of λ for the $(1,\lambda)$ EA leading to an asymptotically optimal runtime on OneMax [26,27]. Kaufmann et al. [28] generalised this results to arbitrary monotone functions.

We argue that providing runtime analyses of parameter control mechanisms for non-elitist EAs is an important direction for research. Non-elitist algorithms are frequently used in practice and understanding the dynamics of non-elitist EAs is vital to narrow the gap between theory and practice. Recent theoretical analysis has shown that non-elitist EAs are able to escape local optima efficiently [29], but they concern only static parameter values. On the other hand, many existing theoretical studies regarding parameter control mechanisms concern elitist EAs optimising fairly easy problems on which algorithms with static parameters already run efficiently, and so the performance gains obtained through parameter control are often moderate at best. More research effort should be devoted to the intersection of non-elitist EAs and parameter control mechanisms on more challenging problems since the potential for performance improvements is much greater.

1.1. Our contribution

In this work we provide an example of significant performance improvements through parameter control for a multimodal problem. We study the $(1,\lambda)$ EA on the multimodal problem Cliff [30] with a mechanism to self-adjust the choice of the offspring population size λ. The function Cliff (formally defined in Section 2) typically requires EAs to jump down a "cliff" by accepting a huge loss in fitness, and then to climb up a slope towards the global optimum. Elitist EAs are unable to accept this fitness loss and typically need to jump directly to the global optimum (Theorem 8 in [31] gives a tight bound for the (1+1) EA). The $(1,\lambda)$ EA is able to jump down the cliff if and only if *all* offspring are generated at the bottom of the cliff. Hence, the smaller the population size, the higher the probability of jumping down the cliff. However, the $(1,\lambda)$ EA also needs to be able to climb up a OneMax-like slope towards the cliff and towards the global optimum. The offspring population size λ must be large enough to enable hill climbing. Rowe and Sudholt [2] showed that there is a phase transition on OneMax at $\log_{\frac{e}{e-1}} n$. More specifically, $\lambda \geq \log_{\frac{e}{e-1}} n$ is sufficient to optimise OneMax efficiently, in expected $O(n\log(n)+\lambda n)$ evaluations, but all $\lambda \le (1-\varepsilon)\log_{\frac{e}{e-1}} n$ lead to exponential optimisation times. Every fixed value of λ must strike a delicate balance to enable jumps down the cliff and at the same time being able to hill climb. Jägersküpper and Storch [30] showed a bound of $O(e^{5\lambda})$ for $\lambda \ge 5\ln n$, which gives an upper bound of $O(n^{25})$ for $\lambda = 5\ln n$. To our knowledge, this is the best known upper bound for the runtime of the $(1,\lambda)$ EA to date. A lower bound of $\min\{n^{n/4}, e^{\lambda/4}\}/3$ for all λ was shown in [30]. Comparing the term $e^{\lambda/4} \approx 1.284^{\lambda}$ to the upper bound of order $e^{5\lambda} \approx 148.413^{\lambda}$, the exponents (to the base of e) differ by a factor of 20 and the bases to the power of λ differ by a factor larger than 115. This leaves a large polynomial gap between upper and lower bounds for $\lambda = \Theta(\log n)$.

We refine the results from [30] and show that the runtime is $O(\lambda\xi^{\lambda}\log n)$ and $\Omega(\lambda\xi^{\lambda})$ for a base of $\xi \approx 6.196878$, for reasonable values of λ. For the best fixed λ, we show that the expected optimisation time grows like n^{η} for a constant $\eta \approx 3.976770136$, up to sub-polynomial factors. More specifically, it is in $O((n/\log n)^{\eta}\log n)$ (that is, it grows slower than n^{η} by at least a polylogarithmic factor), and it grows faster than any polynomial of degree less than η.

More importantly, we then show that parameter control is highly beneficial in this scenario. We present a self-adjusting $(1,\lambda)$ EA that self-adjusts λ and prove that it is able to optimise Cliff in $O(n)$ expected generations and $O(n\log n)$ expected evaluations. This is faster than any static parameter choice by a factor of $\Omega(n^{2.9767})$ and it is asymptotically the best possible runtime for any unary unbiased black-box algorithm [32].

We remark that this is the first bound of $O(n\log n)$ for a standard evolutionary algorithm on Cliff; previously, $O(n\log n)$ bounds were only achieved by using additional mechanisms such as ageing [33] and hyper-heuristics [25].

Our analysis builds on our previous work [26] that analysed a similar algorithm on the simple OneMax function. The considered algorithm works using a variant of the famous one-fifth success rule: in a generation in which the current fitness is increased (a success), λ is decreased by dividing by a factor of $F > 1$, where F is a parameter. In an unsuccessful generation, λ is increased by multiplying with a factor of $F^{1/s}$. The parameter $s > 0$ is called the *success rate* and it implies that, if on average one in $s+1$ generations is successful, the current value of λ is maintained (as we have one success and s unsuccessful generations and so $\lambda_{t+s+1} = \lambda_t \cdot (F^{1/s})^s \cdot 1/F = \lambda_t$).

In [26,27] we showed that with a success rate of $0 < s < 1$ the self-adjusting $(1,\lambda)$ EA optimises OneMax in $O(n)$ expected generations and $O(n\log n)$ expected evaluations. We also showed that larger parameters lead to exponential times. Hence, we use the same restriction $0 < s < 1$ in this work.

To make the self-adjusting $(1,\lambda)$ EA work in multimodal optimisation, we need to tackle an important challenge that requires a redesign of the self-adjusting $(1,\lambda)$ EA in [26]. Success-based parameter control mechanisms can be problematic on multimodal problems because once a local optimum is reached the success of previous generations does not give a good indication of what parameters are needed to escape the local optimum. Strategies have been proposed and analysed to solve this problem, showing a good performance. Some examples from other contexts are: systematically increasing the mutation rate once the neighbourhood has been searched in order to increase the radius of exploration [20,21] or resetting the parameter once it has reached a certain value, allowing the algorithm to cycle through the possible parameter values [11].

We enhance the self-adjusting $(1,\lambda)$ EA from [26] with a resetting mechanism: whenever λ exceeds a predefined maximum of $\lambda_{\max}$, it is reset to $\lambda = 1$. When such a reset happens at the top of the cliff, there is a good probability of jumping down the cliff.

Note that the resetting mechanism may have unwanted side effects: resets may happen in difficult fitness levels, for instance on Cliff resets may happen when climbing up the slope to the global optimum and successes become rare. Hence we need to choose $\lambda_{\max}$ sufficiently large to mitigate this risk and enhance the analysis of the self-adjusting $(1,\lambda)$ EA on OneMax with additional arguments.

1.2. Outline

The paper is structured as follows. Section 2 gives necessary definitions and bounds transition probabilities. Since the current best known upper and lower bounds for the $(1,\lambda)$ EA with static λ on Cliff from [30] are far from tight, we first provide refined upper and lower bounds that are tight up to a logarithmic factor in Section 3. Our upper and lower bounds are then used to determine the degree η of the polynomial term in the runtime for the best fixed value of λ.

In Section 4 we show that despite the possibility of resetting λ near the optimum, the self-adjusting $(1,\lambda)$ EA is able to optimise Cliff in expected $O(n)$ generations and $O(n\log n)$ expected evaluations. We divide the optimisation in several phases showing that the algorithm is able to hill-climb effectively to both the local and global optimum. When it encounters the local optimum, λ typically increases to its maximum, increasing its selection pressure and behaving like an elitist EA. But then λ is reset to 1, reducing the selection pressure of the algorithm allowing it to escape local optima. The behaviour in the local optima is similar to the behaviour of the meta-heuristic from [25] where the algorithm changes from elitism to non-elitism to jump out of local optimum and later on it behaves roughly as a purely elitist algorithm.

In Section 5 we provide a detailed empirical study of the proposed algorithm on Cliff and several other problems. We compare the performance of the $(1,\lambda)$ EA with static parameter values and with self-adjusting mechanism on Cliff with realistic problem sizes. These experiments show that even for small problem sizes the self-adjusting mechanism outperforms static parameter values. Additionally, we explore how the hyperparameter $\lambda_{\max}$ affects the performance of the self-adjusting $(1,\lambda)$ EA on Cliff and OneMax. We show that, as predicted from our analysis, decreasing the value of $\lambda_{\max}$ decreases the runtime on Cliff, but this is true only up to a point where resets become very common and the performance of the algorithm deteriorates. Finally, we compare the $(1,\lambda)$ EA with static and self-adjusting parameters on four combinatorial problems, showing that our results can extend to more complicated fitness

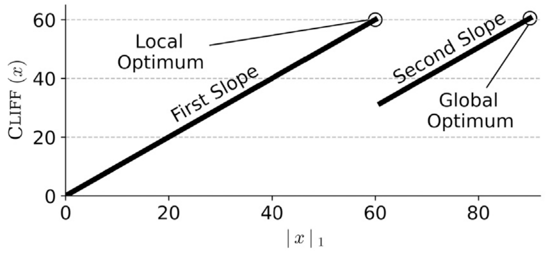

Fig. 1. CLIFF(x) with $n = 90$.

landscapes: the $(1, \lambda)$ EA using the self-adjusting mechanism outperforms static parameter values in three out of the four problems tested.

A preliminary version of our results without the empirical results has been published at a conference [34]. In this extended version we used a recent refinement of the threshold value for efficient optimisation of ONEMAX for static λ from $\lambda \geq \log_{\frac{e}{e-1}} n$ to $\lambda \geq \log_{\frac{e}{e-1}} \left(\frac{cn}{\lambda}\right)$ for some constant $c > e^2$ presented in [35]. This improves our previous upper bounds by a factor of $\log^\eta n$ (i.e. roughly $\log^4 n$). Implementing this improvement required substantial changes in the proofs of Lemma 3.3 and the proof of Theorem 3.1. Also note that Theorems 3.1 and 3.5 in [34] erroneously missed a factor of λ in the stated upper bounds; this issue is fixed in this manuscript.

2. Preliminaries

We present a runtime analysis of the self-adjusting $(1, \lambda)$ EA[1] on the n-dimensional pseudo-Boolean benchmark function CLIFF. We write $|x|_1$ to denote the number of one-bits in the bit string x. By log we refer to the logarithm with base 2.

2.1. *The cliff function class*

The CLIFF benchmark function was first proposed by Jägersküpper and Storch [30] as an example where non-elitism helps the optimisation process. The function is designed to guide hill-climbing algorithms towards a local optimum (*cliff*) where the global optimum is the only other search point with a higher fitness value but it is at a linear distance from the local optimum. An elitist algorithm then needs to perform a large jump to find the global optimum; a non-elitist algorithm instead can escape the local optimum by performing a fitness-decreasing step that leads to another slope guiding to the global optimum. An instance of CLIFF with $n = 90$ is shown in Fig. 1. We define the CLIFF function as follows:

$$\text{CLIFF}(x) := \begin{cases} |x|_1 & \text{if } |x|_1 \leq 2n/3, \\ |x|_1 - n/3 + 1/2 & \text{otherwise.} \end{cases}$$

Following Lissovoi et al. [25] our definition of CLIFF differs from [30] in that the cliff is located at $2n/3$ one-bits instead of $2n/3 - 1$. We choose this definition because it resembles the definition of the JUMP class functions and improves the readability of the runtime analysis. Throughout the analysis we assume n is divisible by 3. Also following [25], we say that all search points x with $|x|_1 \leq 2n/3$ form the *first slope* and all other search points form the *second slope*.

The CLIFF function was used as a benchmark in several other works, including studies of the Strong Selection Weak Mutation (SSWM) model of evolution [31], artificial immune systems [33] and hyper-heuristics [25].

2.2. *The self-adjusting* $(1, \lambda)$ *EA*

The self-adjusting $(1, \lambda)$ EA was first proposed in [26] and studied on ONEMAX. The algorithm uses the generalised success based rule (one-$(s + 1)$-th success rule) to adjust the offspring population size λ. 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*.

In this work we consider a variation of the self-adjusting $(1, \lambda)$ EA with a resetting mechanism for λ (see Algorithm 1) where λ is reset to 1 if $\lambda = \lambda_{\max}$ and there is an unsuccessful generation. This strategy has also been successfully applied to the self-adjusting mechanism of the $(1 + (\lambda, \lambda))$ GA in [11]. In addition, the strategy is similar to the stagnation detection from [20,21] in that if $\lambda_{\max}$ is set appropriately when $\lambda = \lambda_{\max}$ the algorithm is likely to be in a local optimum and the behaviour of the algorithm changes. In this case when λ is large enough the algorithm maintains its fitness with high probability, but when λ is reset to 1 the behaviour changes to a pure random walk, allowing the algorithm to escape local optima.

The algorithm generates new offspring via *standard bit mutations*, which flip each bit independently with probability $1/n$.

[1] named $(1, \{F^{1/s}\lambda, \lambda/F\})$ EA in [26].

Algorithm 1: Self-adjusting $(1, \lambda)$ EA resetting λ.

```
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 $i \in \{1, \dots, \lfloor \lambda \rceil\}$ do
4         $y'_i \in \{0,1\}^n \leftarrow$ standard bit mutation $(x)$;
5     Selection: Choose $y \in \{y'_1, \dots, y'_{\lfloor \lambda \rceil}\}$ with $f(y) = \max\{f(y'_1), \dots, f(y'_{\lfloor \lambda \rceil})\}$ u.a.r.;
6     Update: $x \leftarrow y$;
7     if $f(y) > f(x)$ then $\lambda \leftarrow \max\{\lambda/F, 1\}$;
8     if $f(y) \le f(x) \wedge \lambda = \lambda_{\max}$ then $\lambda \leftarrow 1$;
9     if $f(y) \le f(x) \wedge \lambda \ne \lambda_{\max}$ then $\lambda \leftarrow \min\{\lambda F^{1/s}, \lambda_{\max}\}$;
```

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 [10] and Hevia Fajardo and Sudholt [26], 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.

In our analysis we define $X_0, X_1, \dots$ 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.

2.3. *Transition probabilities*

We now define and estimate transition probabilities that apply to all $(1, \lambda)$ EA variants (with or without self-adjustment) in the context of OneMax and Cliff.

Definition 2.1. For all $\lambda \in \mathbb{N}$ we state the following definitions from [26] in the context of OneMax:

$$p^+_{i,\lambda} = \Pr\left(|x_{t+1}|_1 > i \mid |x_t|_1 = i\right)$$
$$p^-_{i,\lambda} = \Pr\left(|x_{t+1}|_1 < i \mid |x_t|_1 = i\right)$$
$$\Delta^-_{i,\lambda} = \mathrm{E}\left(i - |x_{t+1}|_1 \mid |x_t|_1 = i \cap |x_{t+1}|_1 < i\right)$$

The following probabilities and expectations are tailored to the Cliff function. This includes probabilities for jumping down the cliff ($p^{\downarrow}_{i,\lambda}$), that is, jumping from the first slope to the second slope, jumping back up the cliff ($p^{\uparrow}_{i,\lambda}$), that is, jumping from the second slope to the first slope, increasing the fitness without jumping back up the cliff ($p^{\rightarrow}_{i,\lambda}$), and decreasing the fitness without jumping down the cliff ($p^{\leftarrow}_{i,\lambda}$).

Definition 2.2. For all $\lambda \in \mathbb{N}$ we define:

$$p^{\downarrow}_{i,\lambda} = \begin{cases} \Pr\left(|x_{t+1}|_1 > 2n/3 \mid 2n/3 \ge |x_t|_1 = i\right) & \text{if } i \le 2n/3 \\ 0 & \text{otherwise.} \end{cases}$$

$$p^{\uparrow}_{i,\lambda} = \begin{cases} \Pr\left(|x_{t+1}|_1 \le 2n/3 \mid 2n/3 < |x_t|_1 = i\right) & \text{if } i > 2n/3 \\ 0 & \text{otherwise.} \end{cases}$$

$$p^{\rightarrow}_{i,\lambda} = \begin{cases} \Pr\left(i < |x_{t+1}|_1 \le 2n/3 \mid |x_t|_1 = i\right) & \text{if } i \le 2n/3, \\ \Pr\left(i < |x_{t+1}|_1 \mid |x_t|_1 = i\right) & \text{otherwise.} \end{cases}$$

$$p^{\leftarrow}_{i,\lambda} = \begin{cases} \Pr\left(|x_{t+1}|_1 < i \mid |x_t|_1 = i\right) & \text{if } i \le 2n/3, \\ \Pr\left(2n/3 < |x_{t+1}|_1 < i \mid |x_t|_1 = i\right) & \text{otherwise.} \end{cases}$$

$$\Delta^{\leftarrow}_{i,\lambda} = \begin{cases} \mathrm{E}\left(i - |x_{t+1}|_1 \mid |x_t|_1 = i \cap |x_{t+1}|_1 < i\right) & \text{if } i \le 2n/3, \\ \mathrm{E}\left(i - |x_{t+1}|_1 \mid |x_t|_1 = i \cap 2n/3 < |x_{t+1}|_1 < i\right) & \text{otherwise.} \end{cases}$$

$$\Delta^{\rightarrow}_{i,\lambda} = \begin{cases} \mathrm{E}\left(|x_{t+1}|_1 - i \mid |x_t|_1 = i \cap i < |x_{t+1}|_1 \le 2n/3\right) & \text{if } i \le 2n/3, \\ \mathrm{E}\left(|x_{t+1}|_1 - i \mid |x_t|_1 = i \cap i < |x_{t+1}|_1\right) & \text{otherwise.} \end{cases}$$

Finally, for $i > 2n/3$,

$$\Delta^{\uparrow}_{i,\lambda} = \mathrm{E}\left(i - |x_{t+1}|_1 \mid 2n/3 < |x_t|_1 = i \cap |x_{t+1}|_1 \le 2n/3\right).$$

The events underlying the probabilities from Definition 2.2 are mutually disjoint. They relate to the probabilities defined for OneMax in [26] as follows:

$$p^{+}_{i,\lambda} = p^{\rightarrow}_{i,\lambda} + p^{\downarrow}_{i,\lambda} \tag{1}$$
$$p^{-}_{i,\lambda} = p^{\leftarrow}_{i,\lambda} + p^{\uparrow}_{i,\lambda} \tag{2}$$

The following lemma collects bounds on the above transition probabilities that will be used throughout the remainder.

Lemma 2.3. *For any* $(1,\lambda)$ *EA on* Cliff*, the quantities from Definition 2.2 are bounded as follows: For all* $i \in \{1,\dots,n\}$*,*

$$p^{\leftarrow}_{i,\lambda} \le p^{-}_{i,\lambda} \le \left(\frac{e-1}{e}\right)^{\lambda} \tag{3}$$
$$\Delta^{\leftarrow}_{i,\lambda} \le \Delta^{-}_{i,\lambda} \le \frac{e}{e-1}. \tag{4}$$

For all $i \in \{2n/3,\dots,n\}$*, letting* $d := i - 2n/3$*,*

$$p^{\uparrow}_{i,\lambda} \le \frac{\lambda (i/n)^d}{d!} \tag{5}$$
$$\Delta^{\uparrow}_{i,\lambda} \le d+1. \tag{6}$$

Finally,

$$p^{\rightarrow}_{i,\lambda} \ge \begin{cases} 1 - \left(1 - \frac{1}{3e}\right)^{\lambda} & \text{if } i < 2n/3, \\ 1 - \left(1 - \frac{n-i}{en}\right)^{\lambda} - p^{\uparrow}_{i,\lambda} & \text{if } i > 2n/3. \end{cases} \tag{7}$$

Proof. The first inequality in (3), $p^{\leftarrow}_{i,\lambda} \le p^{-}_{i,\lambda}$, follows from (2). The stated upper bound on $p^{-}_{i,\lambda}$ was shown in [26].

We have $\Delta^{\leftarrow}_{i,\lambda} \le \Delta^{-}_{i,\lambda}$ since for $i \le 2n/3$, $\Delta^{\leftarrow}_{i,\lambda} = \Delta^{-}_{i,\lambda}$ by definition of $\Delta^{\leftarrow}_{i,\lambda}$ and otherwise $\Delta^{\leftarrow}_{i,\lambda} \le \Delta^{-}_{i,\lambda}$ since the definition of $\Delta^{\leftarrow}_{i,\lambda}$ includes the additional condition $|x_{t+1}|_1 > 2n/3$, that is, only jumps to closer targets are considered and longer jumps to the first slope are disregarded. The second inequality in (4) was shown in [26].

To bound $p^{\uparrow}_{i,\lambda}$, we argue that a necessary requirement for creating an offspring with at most $2n/3$ ones is that d one-bits flip. There are $\binom{i}{d}$ ways for choosing d one-bits and the probability that d specific bits are flipped is $(1/n)^d$. (Note that we make no assumption on the remaining $n-d$ bits; in particular, some of them might flip as well.) Thus,

$$p^{\uparrow}_{i,1} \le \binom{i}{d}\left(\frac{1}{n}\right)^d \le \frac{i^d}{d!}\cdot\left(\frac{1}{n}\right)^d \le \frac{(i/n)^d}{d!}.$$

Using the union bound over all offspring, we get

$$p^{\uparrow}_{i,\lambda} \le \frac{\lambda (i/n)^d}{d!}.$$

Now we bound $\Delta^{\uparrow}_{i,\lambda}$. Let $d := i - 2n/3$. In order for an offspring to jump up the cliff it needs to flip at least d one-bits plus the number of zero-bits flipped, let this quantity be k. Let B denote the random number of flipping one-bits in a standard bit mutation with mutation probability $1/n$, then using Lemma 1.7.3 from [36] we get $\mathrm{E}(B \mid B \ge k) \le k + \mathrm{E}(B) = k+1$. Therefore, $\Delta^{\uparrow}_{i,\lambda} \le d+1$. Increasing the offspring population size does not increase $\Delta^{\uparrow}_{i,\lambda}$ because if multiple offspring jump up the cliff, the algorithm will transition to an offspring with a maximum number of ones on the first slope.

To bound $p^{\rightarrow}_{i,\lambda}$ we argue that, for all $i < 2n/3$, if there is an offspring with $i+1$ ones then the number of ones increases. For $|x_t|_1 < 2n/3$ the probability that an offspring flips only one 0-bit is

$$p^{+}_{i,1} \ge \frac{n-i}{n}\left(1-\frac{1}{n}\right)^{n-1} \ge \frac{n-i}{en}. \tag{8}$$

The probability that any of the λ offspring flips only one 0-bit is

$$1 - \left(1 - p^{+}_{i,1}\right)^{\lambda} \ge 1 - \left(1 - \frac{n-i}{en}\right)^{\lambda} \ge 1 - \left(1 - \frac{1}{3e}\right)^{\lambda}.$$

This proves the claimed lower bound on $p^{\rightarrow}_{i,\lambda}$ if $|x_t|_1 < 2n/3$. If $|x_t|_1 > 2n/3$ then there is an offspring with $i+1$ ones with probability

$$1 - \left(1 - \frac{n-i}{en}\right)^{\lambda}.$$

In this case either the number of ones increases or the algorithm goes back up the cliff. Hence, $p^{\rightarrow}_{i,\lambda} + p^{\uparrow}_{i,\lambda} \ge 1 - \left(1 - \frac{n-i}{en}\right)^{\lambda}$ or, equivalently,

$$p^{\rightarrow}_{i,\lambda} \ge 1 - \left(1 - \frac{n-i}{en}\right)^{\lambda} - p^{\uparrow}_{i,\lambda}. \quad \square$$

We also give a lower bound for a mutation creating an offspring from the top of the cliff (that is, a parent with $2n/3$ ones) that increases the number of ones by at least $\frac{c \log\log n}{\log\log\log n}$, for an arbitrary constant $c > 0$. We shall see later that jumps of this length are sufficient to escape from the set of local optima at the top of the cliff for good.

Lemma 2.4. *For every constant $c > 0$ and all $n \geq 2^{2^{2^{3c}}}$ the probability that a standard bit mutation of a search point with $2n/3$ ones yields an offspring with at least $2n/3 + \frac{c \log\log n}{\log\log\log n}$ ones is at least $e^{-1}(\log n)^{-c}$.*

Proof. Let $\kappa := \frac{c \log\log n}{\log\log\log n}$, then a sufficient condition for the sought event is that κ 0-bits flip and no 1-bit flips. Since there are $\binom{n/3}{\kappa}$ ways to choose the κ 0-bits flipping bits, the probability is at least

$$\left(1-\frac{1}{n}\right)^{2n/3}\binom{n/3}{\kappa}\left(\frac{1}{n}\right)^{\kappa} \geq \left(1-\frac{1}{n}\right)^{n-1}\left(\frac{n/3}{\kappa}\right)^{\kappa}\left(\frac{1}{n}\right)^{\kappa} \geq \left(\frac{1}{e}\right)\left(\frac{1}{3\kappa}\right)^{\kappa} = e^{-1}\,(3\kappa)^{-\kappa}.$$

Plugging in κ in the second factor, we get

$$(3\kappa)^{-\kappa} = \left(\frac{3c\log\log n}{\log\log\log n}\right)^{-\frac{c\log\log n}{\log\log\log n}} = 2^{-\frac{c\log\log n}{\log\log\log n}\cdot\log\left(\frac{3c\log\log n}{\log\log\log n}\right)}.$$

By assumption on n, we have $\log\log\log n \geq 3c$, thus $\frac{3c\log\log n}{\log\log\log n} \leq \log\log n$ and we bound the sought probability by

$$e^{-1}2^{-\frac{c\log\log n}{\log\log\log n}\cdot\log\log\log n} = e^{-1}2^{-c\log\log n} = e^{-1}(\log n)^{-c}. \qquad \square$$

In the remainder, we sometimes tacitly use the following argument. If in an iterative process there is an event that happens independently in each step with probability at most p, the probability that the event happens during a phase of T steps, T a random variable with $\mathrm{E}(T) < \infty$, is at most

$$\sum_t \Pr(T=t)\cdot tp = p\cdot \mathrm{E}(T). \tag{9}$$

3. Static parameter settings

We first consider the performance of the $(1,\lambda)$ EA with a static choice of λ. The main result in this section is the following theorem that gives upper and lower bounds for the expected optimisation time of the $(1,\lambda)$ EA on Cliff.

Theorem 3.1. *The expected optimisation time $E(T)$ of the $(1,\lambda)$ EA with static λ on Cliff is*

$$\begin{aligned} &E(T) = \Omega\left(\lambda\xi^{\lambda}\right) && \text{if } \lambda = O(n) \text{ and}\\ &E(T) = O\left(\lambda\xi^{\lambda}\cdot\log n\right) && \text{if } \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right) \leq \lambda = O(\log n), \end{aligned}$$

where $\xi^{-1} := \frac{1}{e}\sum_{a=0}^{\infty}\sum_{b=a+1}^{\infty}\left(\frac{2}{3}\right)^a\left(\frac{1}{3}\right)^b\frac{1}{a!b!} \approx 0.1613715804$, thus $\xi \approx 6.196878$, and $c > e^2$ is a constant.

The lower bound is exponential in λ for all $\lambda = O(n)$. The constant ξ^{-1} roughly represents the probability of increasing the number of ones in a mutation of a parent at the top of the cliff, i.e. a parent with $2n/3$ ones. For $\lambda \leq (1-\varepsilon)\log_{\frac{e}{e-1}} n$, that is, if the offspring population size is too small to allow for hill climbing on OneMax, a much stronger lower bound of $2^{cn^{\varepsilon/2}}$, for some constant $c > 0$, was shown in [2] for all functions with a unique optimum. In the (arguably more interesting) parameter regime $\lambda > (1-\varepsilon)\log_{\frac{e}{e-1}} n$ our result improves upon the only other lower bound we are aware of: [30] showed a lower bound for all λ of $\min\{n^{n/4}, e^{\lambda/4}\}/3$. The term $e^{\lambda/4}$ is roughly 1.284^{λ} and hence considerably lower than our lower bound of $\lambda\cdot 6.196^{\lambda}$. In this parameter regime our lower bound is nearly tight: for the best known values of λ for OneMax [2], the upper bounds only differ from the lower bound by a logarithmic factor.

The condition on λ in the upper bound, $\lambda \geq \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)$ is somewhat awkward to handle since λ appears on both sides of the inequality. In the next lemma we give a simplified bound for this condition that will be useful later on. The lemma requires $\lambda \leq cn^{2/3}$ for a constant $c > 0$ and this condition is satisfied if $\lambda = O(\log n)$ (cf. Theorem 3.1) and n is sufficiently large.

Lemma 3.2. *For all constants $c > e^2$, if $\lambda \leq cn^{2/3}$ then $\left\lceil \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)\right\rceil \geq \log(n)/2$.*

Proof. We use that $\lambda \leq cn^{2/3}$ and that $\log_{\frac{e}{e-1}}(x) > 3\log(x)/2$ for all $x > 1$.

$$\left\lceil \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)\right\rceil \geq \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right) \geq \log_{\frac{e}{e-1}}\left(\frac{n}{n^{2/3}}\right) = \log_{\frac{e}{e-1}}(n)/3 \geq \log(n)/2. \qquad \square$$

To prove Theorem 3.1, we need to show that, for all search points with at most $3n/4$ ones, improvements are found easily if $\lambda \geq \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)$ for some constant $c > e^2$. Note that the considered set of search points includes search points on the first slope as well as search points on the second slope at a linear distance past the top of the cliff. We will see that, once a search point with at least $3n/4$ ones has been reached, the algorithm will not jump back up the cliff, with high probability. The choice of the constant $3/4$ is somewhat arbitrary; we could have chosen any other constant in $(2/3, 1)$ as any such constant implies a linear distance to the top of the cliff.

Lemma 3.3. *Consider the $(1,\lambda)$ EA with $\log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right) \leq \lambda = O(\log n)$ and a current search point x_t with $|x_t|_1 \leq 3n/4$. Then the following statements hold.*

1. *The probability of creating an offspring with $|x_t|_1 + 1$ ones is at least $1 - n^{-0.06}$.*
2. *For all x_t with $|x_t|_1 \in [0, 3n/4] \setminus [2n/3, 2n/3 + \log n]$, the drift in the number of ones is $E\left(|x_{t+1}|_1 - |x_t|_1 \mid x_t\right) \geq 1 - o(1)$.*
3. *For every non-negative $\kappa = o(n^{0.06})$, the expected number of generations until a search point with exactly $2n/3$ ones or at least $2n/3 + \kappa$ ones is reached, starting from a search point x_0 is $O(2n/3 - |x_0|_1)$ if $|x_0|_1 \leq 2n/3$ and $O(\kappa + \log n)$ otherwise.*

Proof. The probability that one fixed offspring has more than $|x_t|_1 + 1$ ones is at least $(n/4)/(en) = 1/(4e)$. By Lemma 3.2, for a sufficiently large n and $\lambda = O(\log n)$ we have $\left\lceil \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)\right\rceil \geq \log(n)/2$. Therefore, the probability that there is an offspring that increases the number of ones is at least

$$1 - \left(1 - \frac{1}{4e}\right)^{\left\lceil \log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)\right\rceil} \geq 1 - \left(1 - \frac{1}{4e}\right)^{\log(n)/2}$$
$$= 1 - \left(\frac{4e}{4e-1}\right)^{-\log_{\frac{4e}{4e-1}}(n)/\left(2\log_{\frac{4e}{4e-1}}(2)\right)} = 1 - n^{-1/\left(2\log_{\frac{4e}{4e-1}}(2)\right)} \geq 1 - n^{-0.06}.$$

For the second statement, let A^{+1} denote the event from the first statement, that is, creating an offspring with $|x_t|_1 + 1$ ones and let $A^{\uparrow}$ be the event that $|x_t|_1 > 2n/3$ and $|x_{t+1}|_1 \leq 2n/3$. By the law of total probability, abbreviating $\Delta := (|x_{t+1}|_1 - |x_t|_1 \mid x_t)$,

$$\mathrm{E}(\Delta) = \underbrace{\mathrm{E}\left(\Delta \mid A^{+1} \cap \overline{A^{\uparrow}}\right) \cdot \Pr\left(A^{+1} \cap \overline{A^{\uparrow}}\right)}_{a} + \underbrace{\mathrm{E}\left(\Delta \mid \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) \cdot \Pr\left(\overline{A^{+1}} \cap \overline{A^{\uparrow}}\right)}_{b} + \underbrace{\mathrm{E}\left(\Delta \mid A^{\uparrow}\right) \cdot \Pr\left(A^{\uparrow}\right)}_{c}.$$

Note that $\Pr\left(A^{+1} \cap \overline{A^{\uparrow}}\right) = 1 - \Pr\left(\overline{A^{+1}} \cup A^{\uparrow}\right) \geq 1 - \Pr\left(\overline{A^{+1}}\right) - \Pr\left(A^{\uparrow}\right) = \Pr\left(A^{+1}\right) - \Pr\left(A^{\uparrow}\right)$ where the inequality is a union bound. Along with the first statement, the a term is at least $1 \cdot \left(\Pr\left(A^{+1}\right) - \Pr\left(A^{\uparrow}\right)\right) \geq 1 - n^{-0.06} - p^{\uparrow}_{i,\lambda}$.

To estimate the b term, we use

$$\mathrm{E}\left(\Delta \mid \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) \cdot \Pr\left(\overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) = \sum_{x=-\infty}^{\infty} x \cdot \Pr\left(\{\Delta = x\} \cap \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) \geq \sum_{x=-\infty}^{-1} x \cdot \Pr\left(\{\Delta = x\} \cap \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right).$$

We split the sum at $x = -\lceil \log n \rceil$ and use $\Pr\left(\{\Delta = x\} \cap \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) \leq \Pr\left(\overline{A^{+1}}\right) \leq n^{-0.06}$ by the first statement to bound summands with negative factors $x \geq -\lceil \log n \rceil + 1$ as follows:

$$\sum_{x=-\lceil \log n \rceil+1}^{-1} x \cdot \Pr\left(\{\Delta = x\} \cap \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) \geq \sum_{x=-\lceil \log n \rceil+1}^{-1} x \cdot n^{-0.06} \geq -(\log n)^2 \cdot n^{-0.06} = -o(1).$$

For the smaller summands $x \leq -\lceil \log n \rceil$ we bound $\Pr(\Delta = x)$ by the probability of flipping at least $|x|$ bits; note that the latter is a necessary condition for the former. The probability of the former event is at most $1/(|x|!)$, thus

$$\sum_{x=-\infty}^{-\lceil \log n \rceil} x \cdot \Pr\left(\{\Delta = x\} \cap \overline{A^{+1}} \cap \overline{A^{\uparrow}}\right) \geq \sum_{x=-\infty}^{-\lceil \log n \rceil} x \cdot \frac{1}{|x|!} = -\sum_{x=-\infty}^{-\lceil \log n \rceil} \frac{1}{|x-1|!}$$

As this series decays exponentially, it is easy to see that the above sum is in $-\Theta(1/((\lceil \log n \rceil - 1)!) = -o(1)$. Together, the b term is in $-o(1)$.

The c term is $-\Delta^{\uparrow}_{i,\lambda} p^{\uparrow}_{i,\lambda}$ by definition. Plugging this together, we get

$$\mathrm{E}(\Delta) \geq 1 - n^{-0.06} - p^{\uparrow}_{i,\lambda} - o(1) - \Delta^{\uparrow}_{i,\lambda} p^{\uparrow}_{i,\lambda} = 1 - n^{-0.06} - (\Delta^{\uparrow}_{i,\lambda} + 1) p^{\uparrow}_{i,\lambda} - o(1).$$

For $|x_t|_1 < 2n/3$, $p^{\uparrow}_{i,\lambda} = 0$ and the claim follows. For $|x_t|_1 > 2n/3 + \log n$, by Lemma 2.3 with $d \geq \log n$,

$$(\Delta^{\uparrow}_{i,\lambda} + 1) p^{\uparrow}_{i,\lambda} \leq \frac{\lambda(d+2)}{d!} \leq \frac{\lambda(\log(n) + 2)}{(\log n)!} = n^{-\omega(1)}.$$

This implies the second statement.

To show the third statement, we define the following distance function:

$$d(x) := \begin{cases} \frac{2n}{3} - |x|_1 & \text{if } |x|_1 \leq 2n/3 \\ \frac{2n}{3} + \kappa - |x|_1 + \log n & \text{if } 2n/3 < |x|_1 < 2n/3 + \kappa \\ 0 & \text{otherwise.} \end{cases}$$

Recall that the probability of flipping at least $\log n$ bits is $n^{-\omega(1)}$, even when taking a union bound over $\lambda = O(\log n)$ mutations in one generation, hence we typically flip less than $\log n$ bits in all mutations of one generation and denote this event as $F^{\leq \log n}$. If $\left|x_t\right|_1 < 2n/3$ then conditional on the event $A^{+1} \cap F^{\leq \log n}$ an offspring with a larger number of ones in the first slope will be accepted. This implies $d(x_{t+1}) \leq d(x_t) - 1$. If $2n/3 < \left|x_t\right|_1 < 2n/3 + \kappa$, conditional on $A^{+1} \cap F^{\leq \log n}$ we either have $\left|x_{t+1}\right|_1 > \left|x_t\right|_1$ and thus $d(x_{t+1}) \leq d(x_t) - 1$ or the algorithm jumps back up the cliff and $\left|x_t\right|_1 - \log n \leq \left|x_{t+1}\right|_1 \leq 2n/3$, which implies $d(x_{t+1}) \leq \log n$. Along with $d(x_t) = 2n/3 + \kappa - \left|x_t\right|_1 + \log n > \kappa + \log n \geq \log n$, we get $d(x_{t+1}) \leq d(x_t) - 1$. Conditional on $\overline{A^{+1}} \cap F^{\leq \log n}$, $d(x_{t+1}) - d(x_t)$ is always bounded by $\kappa + 2\log n$ since the number of ones changes by at most $\log n$ and the distance increases by $\kappa + \log n$ by definition of the distance function d when crossing from the first slope to the second slope. If more than $\log n$ bits flip, we use the trivial bound $d(x_{t+1}) - d(x_t) \leq n$.

Together, we established the following drift bound. For all x_t with $d(x_t) > 0$, recalling $\kappa = o(n^{0.06})$,

$$\begin{aligned} \mathrm{E}\left(d(x_t) - d(x_{t+1}) \mid x_t\right) &\geq \Pr\left(A^{+1} \cap F^{\leq \log n}\right) \cdot 1 - \Pr\left(\overline{A^{+1}} \cap F^{\leq \log n}\right) \cdot (\kappa + 2\log n) - \Pr\left(\overline{F^{\leq \log n}}\right) \cdot n \\ &\geq (1 - n^{-0.06} - n^{-\omega(1)}) \cdot 1 - n^{-0.06} \cdot (\kappa + 2\log n) - n^{-\omega(1)} \cdot n \\ &= 1 - o(1). \end{aligned}$$

Since the initial distance is $2n/3 - \left|x_0\right|_1$ if $\left|x_0\right|_1 \leq 2n/3$ and at most $\kappa + \log n$ otherwise, the claim follows from the additive drift theorem [37]. □

Another important step for proving Theorem 3.1 is estimating the probability of a standard bit mutation of a parent at the top of the cliff increasing the number of ones, $p^{+}_{2n/3,1}$. This is because in order to jump down the cliff, all offspring must increase the number of ones, which has a probability of $(p^{+}_{2n/3,1})^{\lambda}$. To prove the claimed upper and lower bounds in Theorem 3.1 we need precise estimations of $p^{+}_{2n/3,1}$ as it appears in the base of an expression exponential in λ; the commonly used inequalities $\frac{n-2n/3}{en} \leq p^{+}_{2n/3,1} \leq \frac{n-2n/3}{n}$ (that is, $\frac{1}{3e} \leq p^{+}_{2n/3,1} \leq \frac{1}{3}$) are too loose. The following lemma gives precise upper and lower bounds on $p^{+}_{i,1}$ for almost all values of i as this generality is achieved quite easily and the lemma may be of independent interest.

Lemma 3.4. *For all $n \geq 2$ and all $i \in \{0, \dots, n-1\}$,*

$$p^{+}_{i,1} \leq \left(1 - \frac{1}{n}\right)^{n-2} \sum_{a=0}^{\infty} \sum_{b=a+1}^{\infty} \left(\frac{i}{n}\right)^{a} \left(\frac{n-i}{n}\right)^{b} \frac{1}{a!b!}. \tag{10}$$

For all $n \geq 2$ and all $i \in \{1, \dots, n-1\}$,

$$p^{+}_{i,1} \geq \left(1 - \frac{1}{n}\right)^{n-2} \sum_{a=0}^{\infty} \sum_{b=a+1}^{\infty} \left(\frac{i}{n}\right)^{a} \left(\frac{n-i}{n}\right)^{b} \frac{1}{a!b!} \left(1 - \frac{2\log^2 n}{\min\{i, n-i\}}\right). \tag{11}$$

Note that the lower bound on $p^{+}_{i,1}$ in Equation (11) is only meaningful for $i \in \{\lceil 2\log^2 n\rceil, \dots, n - \lceil 2\log^2 n\rceil\}$ as otherwise $1 - \frac{2\log^2 n}{\min\{i,n-i\}} \leq 0$. This implies that the right-hand side is negative or 0 and the claim is trivial. We exclude $i = 0$ in (11) to avoid division by $\min\{0, n-0\} = 0$.

Proof of Lemma 3.4. Let f^{ℓ}_{k} denote the probability of flipping k bits out of a set of ℓ bits. We have an increase in the number of ones if more zeros flip than ones. Since mutation treats ones and zeros independently,

$$p^{+}_{i,1} = \sum_{a=0}^{\infty} \sum_{b=a+1}^{\infty} f^{i}_{a} \cdot f^{n-i}_{b}. \tag{12}$$

(We let these sums run to ∞ to ease notation, but only finitely many terms with $a \leq i$ and $b \leq n-i$ are non-zero.)

We bound f^{ℓ}_{k} from above and below to show the claim. For all $1 \leq k \leq \ell \leq n$,

$$\begin{aligned} f^{\ell}_{k} &= \binom{\ell}{k}\left(\frac{1}{n}\right)^{k}\left(1 - \frac{1}{n}\right)^{\ell-k} \\ &= \frac{\ell(\ell-1)\cdot \ldots \cdot(\ell-k+1)}{k!}\left(\frac{1}{n}\right)^{k}\left(1 - \frac{1}{n}\right)^{\ell-k} \end{aligned}$$

$$\begin{aligned}
&\le \frac{\ell(\ell-1)^{k-1}}{k!}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-k}\\
&\le \frac{\ell(\ell-\ell/n)^{k-1}}{k!}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-k}\\
&= \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}.
\end{aligned}$$

For $k=0$ we have $f_k^\ell = \left(1-\frac{1}{n}\right)^\ell \le \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}$ as well. Plugging this into (12) establishes the claimed upper bound.

To bound $p_{i,1}^+$ from below, we drop summands from (12) where $a > \log n$ or $b > \log n$, that is, more than $\log n$ bits flip in one mutation, and obtain:

$$p_{i,1}^+ \ge \sum_{a=0}^{\log n}\sum_{b=a+1}^{\log n} f_a^i \cdot f_b^{n-i}. \tag{13}$$

Now, for $1 \le k \le \log n$ and $k \le \ell \le n$,

$$\begin{aligned}
f_k^\ell &= \binom{\ell}{k}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-k}\\
&= \frac{\ell(\ell-1)\cdot\ldots\cdot(\ell-k+1)}{k!}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-k}\\
&\ge \frac{(\ell-\log n)^k}{k!}\left(\frac{1}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}\\
&= \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}\left(1-\frac{\log n}{\ell}\right)^k\\
&\ge \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}\left(1-\frac{k\log n}{\ell}\right)\\
&\ge \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}\left(1-\frac{\log^2 n}{\ell}\right).
\end{aligned}$$

For $k=0$ and $\ell \ge 1$ we have

$$\begin{aligned}
f_k^\ell &= \left(1-\frac{1}{n}\right)^\ell = \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}\left(1-\frac{1}{n}\right)\\
&\ge \frac{1}{k!}\left(\frac{\ell}{n}\right)^k\left(1-\frac{1}{n}\right)^{\ell-1}\left(1-\frac{\log^2 n}{\ell}\right).
\end{aligned}$$

Plugging this into Equation (13) (once for $k=a$ and $\ell=i$ and again for $k=b$ and $\ell=n-i$) and simplifying $\left(1-\frac{1}{n}\right)^{i-1}\cdot\left(1-\frac{1}{n}\right)^{n-i-1} = \left(1-\frac{1}{n}\right)^{n-2}$,

$$p_{i,1}^+ \ge \left(1-\frac{1}{n}\right)^{n-2}\sum_{a=0}^{\log n}\sum_{b=a+1}^{\log n}\frac{1}{a!}\left(\frac{i}{n}\right)^a\left(1-\frac{\log^2 n}{i}\right)\frac{1}{b!}\left(\frac{n-i}{n}\right)^b\left(1-\frac{\log^2 n}{n-i}\right).$$

Noting that

$$\left(1-\frac{\log^2 n}{i}\right)\left(1-\frac{\log^2 n}{n-i}\right) \ge 1-\frac{2\log^2 n}{\min\{i,n-i\}}$$

completes the proof. □

For the specific value $i=2n/3$, we obtain the following special case. Along with $\frac{1}{e} \le \left(1-\frac{1}{n}\right)^{n-2} \le \frac{1}{e}\cdot(1+O(1/n))$, Equations (10) and (11) in Lemma 2.3 imply the following.

Corollary 3.5.

$$p_{2n/3,1}^+ = \frac{1}{e}\sum_{a=0}^{\infty}\sum_{b=a+1}^{\infty}\left(\frac{2}{3}\right)^a\left(\frac{1}{3}\right)^b\frac{1}{a!b!} \pm O\left(\frac{\log^2 n}{n}\right)$$

which is approximately $0.1613715804 \pm O(\log^2(n)/n)$.

Now we are ready to prove Theorem 3.1.

Proof of Theorem 3.1. By Chernoff bounds, with probability $1 - e^{-\Omega(n)}$, the initial search point has at most $2n/3$ ones. In the following, we assume that in the first $\Theta(\xi^\lambda)$ expected generations, no mutation ever flips at least $n/3$ bits. The probability of flipping at least $n/3$ bits in one mutation is at most $1/((n/3)!) = n^{-\Omega(n)}$ and a union bound over λ offspring and $\Theta(\xi^\lambda)$ expected generations (cf. (9)) still yields a probability of $n^{-\Omega(n)}$.

Under this assumption, a necessary condition for finding the optimum is that a transition from a search point with at most $2n/3$ ones to a search point with more than $2n/3$ ones is made (i.e. a jump down the cliff). By Lemma 1 in [38], the probability of this event is maximised if the parent has exactly $2n/3$ ones; then it is $(p^+_{2n/3,1})^\lambda$. By Equation (10) in Lemma 3.4, along with $(1-1/n)^{n-2} \le 1/e \cdot (1-1/n)^{-2} \le 1/e \cdot (1+2/n)$,

$$(p^+_{2n/3,1})^\lambda \le \left(\xi^{-1}\left(1+\frac{2}{n}\right)\right)^\lambda \le \xi^{-\lambda} \cdot e^{2\lambda/n} = O(\xi^{-\lambda}).$$

The expected waiting time for this transition to happen is at least $\Omega(\xi^\lambda)$. This establishes a lower bound for the number of generations of $(1 - e^{-\Omega(n)} - n^{-\Omega(n)}) \cdot \Omega(\xi^\lambda) = \Omega(\xi^\lambda)$. Multiplying the expected number of generations by λ yields the claimed lower bound for the number of evaluations.

Now we show the upper bound. We consider the number of generations T_κ until a search point with at least $2n/3 + \kappa$ ones is found, for $\kappa := \frac{2\log\log n}{\log\log\log n}$, assuming that the current search point is at the top of the cliff, that is, the current search point has $2n/3$ ones.

Let A^- denote the event of accepting a search point with less than $2n/3$ ones from the top of the cliff and let A^+ denote the event of accepting a search point with more than $2n/3$ ones from the top of the cliff. Note that $\Pr\left(A^+\right) = (p^+_{2n/3,1})^\lambda$ as all offspring must have a larger number of ones (we tacitly ignore the possibility of reaching the global optimum via a direct jump).

By Lemma 2.4 with $c := 2$, the probability of creating an offspring at distance at least $\kappa := \frac{2\log\log n}{\log\log\log n}$ from the cliff is at least $1/(e\log^2 n)$. The probability that there is one such offspring is at least

$$1 - \left(1 - \frac{1}{e\log^2 n}\right)^\lambda \ge \frac{\lambda/(e\log^2 n)}{1+\lambda/(e\log^2 n)} \ge \frac{\lambda}{2e\log^2 n} \ge \frac{1}{6\log n},$$

where the first inequality follows from Lemma 10 in [39] and the last inequality follows from Lemma 3.2 and the assumption $\log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right) \le \lambda = O(\log n)$. We denote this event as A^{jump} and note that in the event of $A^+ \cap A^{\text{jump}}$ a search point with at least $2n/3 + \kappa$ ones is accepted. The event A^+, all offspring having more than $2n/3$ ones, facilitates the event A^{jump} of a jump, and the event of a jump facilitates the event A^+. Hence we can bound the probability of the intersection of these events by their products:

$$\Pr\left(A^+ \cap A^{\text{jump}}\right) \ge \Pr\left(A^+\right) \cdot \Pr\left(A^{\text{jump}}\right) \ge \frac{\Pr\left(A^+\right)}{6\log n}. \tag{14}$$

In case $A^+ \cap A^{\text{jump}}$ does not occur, the algorithm remains in the local optimum or, in case it has been left, will return to a local optimum eventually. Let T_{return} denote the expected number of iterations needed to return to a local optimum (or a search point with at least $2n/3 + \kappa$ ones), starting with a search point x_0 that resulted from an iteration that left a local optimum.

We recall the event $F^{\le \log n}$ of flipping at most $\log n$ bits in all λ mutations from the proof of Lemma 3.3. If, when leaving a local optimum, $F^{\le \log n}$ occurred, we have $|x_0|_1 \le 2n/3 - \log n$ or $|x_0|_1 > 2n/3$. In both cases, Lemma 3.3 implies $\mathrm{E}\left(T_{\text{return}} \mid F^{\le \log n}\right) = O(\kappa + \log n) = O(\log n)$. Bounding $\mathrm{E}\left(T_{\text{return}} \mid \overline{F^{\le \log n}}\right) = O(n)$ by Lemma 3.3 and $\Pr\left(\overline{F^{\le \log n}}\right) = n^{-\omega(1)}$, by the law of total probability the unconditional expectation is

$$\begin{aligned}\mathrm{E}\left(T_{\text{return}}\right) &= \Pr\left(F^{\le \log n}\right)\mathrm{E}\left(T_{\text{return}} \mid F^{\le \log n}\right) + \Pr\left(\overline{F^{\le \log n}}\right)\mathrm{E}\left(T_{\text{return}} \mid \overline{F^{\le \log n}}\right)\\ &\le O(\log n) + n^{-\omega(1)} \cdot O(n) = O(\log n).\end{aligned}$$

We now establish a recurrence for $\mathrm{E}\left(T_\kappa\right)$ using the above arguments. Starting with a search point at the top of the cliff, the algorithm executes one generation and then makes one of the following transitions. In the event $A^+ \cap A^{\text{jump}}$ the goal is reached. In case of $(A^+ \cup A^-) \setminus (A^+ \cap A^{\text{jump}})$, the algorithm continues from a location other than the top of the cliff. Here we wait until the algorithm returns to the top of the cliff (or to a search point with at least $2n/3 + \kappa$ ones), which takes expected time $\mathrm{E}\left(T_{\text{return}}\right)$, and then the remaining expected time is given by $\mathrm{E}\left(T_\kappa\right)$. In all other cases, the algorithm remains at the top of the cliff and the remaining expected time is $\mathrm{E}\left(T_\kappa\right)$. Together,

$$\mathrm{E}\left(T_\kappa\right) \le 1 + \Pr\left((A^+ \cup A^-) \setminus (A^+ \cap A^{\text{jump}})\right) \cdot \left(\mathrm{E}\left(T_{\text{return}}\right) + \mathrm{E}\left(T_\kappa\right)\right) + \left(1 - \Pr\left(A^+ \cup A^-\right)\right) \cdot \mathrm{E}\left(T_\kappa\right).$$

Subtracting $(\Pr\left((A^+ \cup A^-) \setminus (A^+ \cap A^{\text{jump}})\right) + \left(1 - \Pr\left(A^+ \cup A^-\right)\right)) \cdot \mathrm{E}\left(T_\kappa\right) = (1 - \Pr\left(A^+ \cap A^{\text{jump}}\right) \cdot \mathrm{E}\left(T_\kappa\right)$ on both sides, we obtain

$$\begin{aligned}\Pr\left(A^+ \cap A^{\text{jump}}\right) \cdot \mathrm{E}\left(T_\kappa\right) &\le 1 + \Pr\left((A^+ \cup A^-) \setminus (A^+ \cap A^{\text{jump}})\right) \cdot \mathrm{E}\left(T_{\text{return}}\right)\\ &\le 1 + \Pr\left(A^+ \cup A^-\right) \cdot \mathrm{E}\left(T_{\text{return}}\right).\end{aligned} \tag{15}$$

We argue that, at the top of the cliff, the probability of moving to a search point with a different number of ones is $O(1/n^{0.18})$. This is because if there is a mutation that does not flip any bits, the next search point will be at the top of the cliff again. Hence, in order to move to a search point with a different number of ones, all offspring must flip at least one bit. The probability of this event is at most, using $(1-1/n)^n \geq 1/4$ for $n \geq 2$ and $1+x \leq e^x$

$$\Pr\left(A^+ \cup A^-\right) \leq \left(1-\left(1-\frac{1}{n}\right)^n\right)^\lambda \leq (1-1/4)^\lambda \leq e^{-\lambda/4} \leq e^{-\log_{\frac{e}{e-1}}\left(\frac{cn}{\lambda}\right)/4} \leq e^{-\log_2(n)/8} \leq n^{-1/(8\ln(2))} = O(1/n^{0.18}).$$

Plugging this and (14) into (15) and multiplying both sides by $\frac{6\log(n)}{\Pr(A^+)}$, we get

$$\mathrm{E}\left(T_\kappa\right) \leq O\left(\frac{\log n}{\Pr\left(A^+\right)}\right).$$

Now assume that a search point with $2n/3+d$ ones has been reached, where $\kappa \leq d \leq \log n$. By Lemma 3.3, Lemma 2.3 and a union bound, the probability that in one generation the number of ones is increased (i.e. the algorithm does not jump up the cliff again) is at least

$$1-n^{-0.06}-\frac{\lambda(3/4)^d}{d!}.$$

By a union bound, the probability that this happens for all $d=\kappa \dots n^{0.03}$ is at least

$$1-\frac{n^{0.03}}{n^{0.06}}-\lambda\sum_{d=\kappa}^{n^{0.03}}\frac{(3/4)^d}{d!}.$$

The sum is bounded from above, using $d! \geq (d/e)^e$, as

$$\sum_{d=\kappa}^{n^{0.03}}\frac{(3/4)^d}{d!} \leq \sum_{d=\kappa}^{n^{0.03}}\left(\frac{3e}{4d}\right)^d \leq \sum_{d=\kappa}^{\infty}\left(\frac{3e}{4\kappa}\right)^d = \left(\frac{3e}{4\kappa}\right)^\kappa \cdot \sum_{d=0}^{\infty}\left(\frac{3e}{4\kappa}\right)^d = \left(\frac{3e}{4\kappa}\right)^\kappa \cdot \frac{1}{1-\frac{3e}{4\kappa}} \leq 2\left(\frac{3e}{4\kappa}\right)^\kappa.$$

Plugging in κ, we get

$$2\left(\frac{3e\log\log\log n}{8\log\log n}\right)^{\frac{2\log\log n}{\log\log\log n}} = 2\cdot 2^{\frac{2\log\log n}{\log\log\log n}\cdot\log\left(\frac{3e\log\log\log n}{8\log\log n}\right)} = 2\cdot 2^{-\frac{2\log\log n}{\log\log\log n}\cdot\log\left(\frac{8\log\log n}{3e\log\log\log n}\right)}.$$

For large enough n, we have

$$\frac{8\log\log n}{3e\log\log\log n} \geq 2(\log\log n)^{1/2}$$

and then

$$\log\left(\frac{8\log\log n}{3e\log\log\log n}\right) \geq 1+\frac{1}{2}\log\log\log n.$$

Plugging this in, we bound the sum by

$$2\cdot 2^{-\frac{2\log\log n}{\log\log\log n}\cdot\left(1+\frac{1}{2}\log\log\log n\right)} \leq 2\cdot 2^{-\log\log n-\frac{2\log\log n}{\log\log\log n}} = o\left(\frac{1}{\log n}\right).$$

Thus, the probability of reaching a search point with at least $2n/3+n^{0.03}$ ones before going back up the cliff is at least

$$1-\frac{n^{0.03}}{n^{0.06}}-\lambda\sum_{d=\kappa}^{\log n}\frac{(3/4)^d}{d!} \geq 1-o(1).$$

From that point on, for any search point with at least $2n/3+n^{0.03}/2$ ones, we can use arguments from the analysis of the $(1,\lambda)$ EA on OneMax in [2] since the $(1,\lambda)$ EA must flip at least $n^{0.03}/2$ bits to return to the cliff, and this has exponentially small probability. It is not difficult to show using the negative drift theorem [40,41] and the second statement of Lemma 3.3, that, once we have reached a distance of at least $n^{0.03}$ from the cliff, reducing that distance to at most $n^{0.03}/2$ has an exponentially small probability. Assuming we never go back up the cliff, the remaining expected optimisation time is $O(n\log n)$ [2].

In total, the expected optimisation time (number of evaluations) is

$$O(1)\cdot\lambda\mathrm{E}\left(T_\kappa\right) = O\left(\frac{\lambda\log n}{\Pr\left(A^+\right)}\right) = O\left(\frac{\lambda\log n}{(p^+_{2n/3,1})^\lambda}\right).$$

This implies the claim since $p^+_{2n/3,1} = \xi^{-1} - O((\log^2 n)/n)$ by Corollary 3.5 and

$$\left(p^{+}_{2n/3,1}\right)^{\lambda} = \left(\xi^{-1} - \frac{O(\log^2 n)}{n}\right)^{\lambda} = \xi^{-\lambda}\left(1 - \frac{O(\log^2 n)}{n}\right)^{\lambda} \geq \xi^{-\lambda}\left(1 - \frac{O(\lambda \log^2 n)}{n}\right) = \Omega(\xi^{-\lambda}). \quad \square$$

Along with the exponential lower bound from [2] for small λ-values, Theorem 3.1 shows that the expected optimisation time for any λ grows faster than any polynomial with degree less than $\eta \approx 3.976770136$.

Theorem 3.6. *Let $\eta := 1/\log_{\xi}\left(\frac{e}{e-1}\right) \approx 3.976770136$. The expected optimisation time of the $(1,\lambda)$ EA with static λ is $\omega(n^{\eta-\varepsilon}\log n)$ for every constant $\varepsilon > 0$ and every λ and $O((n/\log n)^{\eta}\log n)$ for $\lambda = \left\lceil \log_{\frac{e}{e-1}}(cn/\lambda)\right\rceil$ with a constant $c > e^2$.*

Proof. By Theorem 3.1, the expected optimisation time for $\lambda = \left\lceil \log_{\frac{e}{e-1}}(cn/\lambda)\right\rceil$ with a constant $c > e^2$ is $O(\xi^{\lambda}\log n)$. Now,

$$\xi^{\lambda} \leq \xi^{1+\log_{\frac{e}{e-1}}(cn/\lambda)} = \xi^{1+\frac{\log_{\xi}(cn/\lambda)}{\log_{\xi}(\frac{e}{e-1})}} = \xi \cdot \left(\frac{cn}{\lambda}\right)^{1/\log_{\xi}(\frac{e}{e-1})} = \xi \cdot \left(\frac{cn}{\lambda}\right)^{\eta}.$$

By Lemma 3.2 $\lambda \geq \frac{\log_2 n}{2}$, hence, this is at most $\xi \cdot \left(\frac{2cn}{\log n}\right)^{\eta}$. Plugging $\xi^{\lambda} \leq \xi \cdot \left(\frac{2cn}{\log n}\right)^{\eta}$ into $O(\xi^{\lambda}\log n)$ establishes the claimed upper bound.

For the lower bound, we exploit that for every constant $\varepsilon' > 0$, for all $\lambda \leq (1-\varepsilon')\log_{\frac{e}{e-1}} n$ the expected optimisation time of the $(1,\lambda)$ EA on every function with a unique optimum is at least $2^{cn^{\varepsilon'}/2}$, for some constant $c > 0$, by Theorem 10 in [2]. This is clearly in $\omega(n^{\eta}\log n)$. For $\lambda = \omega(n)$ the lower bound $\min\{n^{n/4}, e^{\lambda/4}\}/3$ from Theorem 8 in [30] is exponential. It thus suffices to consider $\lambda > (1-\varepsilon')\log_{\frac{e}{e-1}} n$ and $\lambda = O(n)$. The lower bound from Theorem 3.1 then becomes $\Omega(\xi^{(1-\varepsilon')\log_{\frac{e}{e-1}} n}\log n) = \Omega(n^{(1-\varepsilon')\eta}\log n)$. Choosing $\varepsilon' := \varepsilon/(2\eta)$, this is $\Omega(n^{\eta-\varepsilon/2}\log n) = \omega(n^{\eta-\varepsilon}\log n)$ as claimed. $\square$

Recall that the degree η of the polynomial in our runtime bounds is determined by the constant ξ defined by $\xi^{-1} := \frac{1}{e}\sum_{a=0}^{\infty}\sum_{b=a+1}^{\infty}\left(\frac{2}{3}\right)^{a}\left(\frac{1}{3}\right)^{b}\frac{1}{a!b!} \approx 0.1613715804$. Further recall that ξ^{-1} roughly corresponds to the probability of increasing the number of ones at the top of the cliff. The constant $2/3$ is derived from the position of the cliff at $(2/3)\cdot n$ ones. The constant $1/3$ is derived from $1-2/3$. If the location of the cliff was changed to another linear value in n with a leading constant in $(1/2, 1)$ our analysis could be adapted with minor modifications and would yield a polynomial of a possibly different degree. Since this work is already quite technical for the specific location at $(2/3)\cdot n$, we leave such a generalisation for future work.

4. Self-adjusting offspring populations are efficient on cliff

In this section we show that the self-adjusting $(1,\lambda)$ EA is faster than the $(1,\lambda)$ EA with the best static parameter choice by a polynomial factor of $\Omega(n^{2.9767})$ (up to sub-polynomial terms), achieving the best possible asymptotic runtime for any unary unbiased black-box algorithm of $O(n\log n)$ [32]. The main result of this section is shown in Theorem 4.1.

Theorem 4.1. *Let the update strength $F > 1$ and the success rate $0 < s < 1$ be constants and $enF^{1/s} \leq \lambda_{\max} = \text{poly}(n)$. Then for any initial search point and any initial $\lambda_0 \leq \lambda_{\max}$ the self-adjusting $(1,\lambda)$ EA resetting λ optimises* Cliff *in $O(n)$ expected generations and $O(\lambda_{\max}\log n)$ expected function evaluations.*

For $\lambda_{\max} = \lceil enF^{1/s}\rceil$ we obtain $O(n\log n)$ evaluations in expectation.

Note that the bound on the expected number of *evaluations* includes a factor of $\lambda_{\max}$ and the bound on the expected number of *generations* is fixed at $O(n)$, regardless of $\lambda_{\max}$. Theorem 4.1 requires $\lambda_{\max} \geq enF^{1/s}$ to avoid frequent undesired resets when approaching the global optimum. Choosing $\lambda_{\max}$ larger than this may increase the expected number of evaluations proportionally to $\lambda_{\max}$. The reason is that then the time for having desirable resets in local optima is increased. Hence picking a $\lambda_{\max}$ larger than the ideal value of $enF^{1/s}$ incurs a relatively small cost, whereas we conjecture that choosing $\lambda_{\max}$ too small may drastically increase the expected number of generations and evaluations as it prevents the algorithm from hill-climbing effectively.

The proof of Theorem 4.1 is divided in four phases: reaching the cliff, jumping down the cliff, climbing away from the cliff and finding the global optimum.

4.1. Reaching the cliff

We note that the algorithm studied here is the same as the algorithm studied in [26,27] as long as all generations that use $\lambda = \lambda_{\max}$ are successful. Here we show that w.o.p. the self-adjusting $(1,\lambda)$ EA does not reset λ before reaching the cliff. By (9) this holds for any random time period of polynomial expected length. The following lemma concerns a larger region of search points with up to $3n/4$ ones as this will be useful later on.

Lemma 4.2. *Consider the self-adjusting* $(1, \lambda)$ *EA as in Theorem 4.1. The probability that in a generation* t *with* $|x_t|_1 \leq 3n/4$ *and* $|x_t|_1 \neq 2n/3$ *the self-adjusting* $(1, \lambda)$ *EA resets* λ *is at most* $e^{-\Omega(n)}$.

Proof. In order to reset λ a generation using $\lambda = \lambda_{\max}$ must not increase the fitness. By Lemma 2.3 with $\lambda = \lambda_{\max} \geq enF^{1/s}$ the probability of this event is at most

$$1 - p^{\rightarrow}_{\lambda_{\max}} \leq \left(1 - \frac{1}{3e}\right)^{enF^{1/s}} \leq \exp\left(-\frac{nF^{1/s}}{3}\right) = e^{-\Omega(n)}. \qquad \square$$

We note that the results from [26,27] on OneMax can be applied when only considering improvements that increase the fitness by 1. Hence, they can be translated to the Cliff function to calculate the time the algorithm takes to reach the cliff, giving the following bounds.

Lemma 4.3 *(Adapted from Lemmas 3.6 and 3.9 in [27]). Consider the self-adjusting* $(1, \lambda)$ *EA as in Theorem 4.1. For every initial offspring population size* $\lambda_0 \leq \lambda_{\max}$ *and every initial search point* x_0 *with* $|x_0|_1 = 2n/3 - a$ *for* $a \in \mathbb{N}$ *the algorithm evaluates a solution* x_t *with* $|x_t|_1 \geq 2n/3$ *using in expectation* $O(a + \log n)$ *generations and* $O(a + \log n + \lambda_0)$ *evaluations.*

Proof of Lemma 4.3. Starting from a solution x_0 with $|x_0|_1 = 2n/3 - a$ the self-adjusting $(1, \lambda)$ EA reaches a solution x_t with $|x_t|_1 \geq 2n/3$ in $O(a + \log n)$ expected generations and $O(\lambda_0) + O(a + \log n) \cdot \frac{1}{p^{+}_{2n/3-1,1}}$ expected evaluations by Lemma 3.6 and 3.9 in [27] respectively (recall from Definition 2.1 that $p^{+}_{2n/3-1,1}$ is the probability of increasing the number of ones from $2n/3 - 1$ ones in one offspring creation). For the bound on the expected number of evaluations we note that by Lemma 2.3 $\frac{1}{p^{+}_{2n/3-1,1}} \leq 3e$. Therefore, $O(\lambda_0) + O(a + \log n) \cdot \frac{1}{p^{+}_{2n/3-1,1}} = O(\lambda_0 + a + \log n)$.

Finally, if the failure from Lemma 4.2 occurs we restart the analysis with a worst-case value of n for a. Since the failure has an exponentially small probability, this does not affect the claimed expectations. $\square$

4.2. Jumping down the cliff

After reaching the cliff, the algorithm needs to jump down the cliff. This requires a generation in which *all* offspring lie on the second slope. We have seen in Section 3 that this probability is exponentially small in λ_t. The resetting mechanism implies that when λ reaches its maximum value $\lambda_{\max}$ and the following generation is unsuccessful, we reach small values of λ_t and jumps down the cliff become likely.

We also know from Section 3 that we need a sufficiently large jump to prevent the algorithm from jumping straight back up the cliff. This probability decreases with the distance to the cliff (that is, $|x_t|_1 - 2n/3$) and it increases with λ_t as many offspring can amplify the probability of a jump back up the cliff. The following definition captures states from which the probability of jumping up the cliff is sufficiently small.

Definition 4.4. Given some even value $\kappa \in \mathbb{N}$, a state (x_t, λ_t) is called *κ-safe* if $|x_t|_1 \geq 2n/3 + \kappa$ and $\lambda_t \leq 2^{-\kappa/2} \cdot (\kappa/2)!$.

Note that $2^{-\kappa/2} \cdot (\kappa/2)!$ is non-decreasing in κ.

In this subsection we give upper bounds on the expected number of generations and the expected number of function evaluations to reach a κ-safe state for the specific value $\kappa := \frac{\log \log n}{\log \log \log n}$. We also consider search points with at least $3n/4$ ones as safe, regardless of the value of λ_t; reaching such a search point will be the goal of the following phase.

Lemma 4.5. *Consider the self-adjusting* $(1, \lambda)$ *EA with resetting* λ *as in Theorem 4.1, with* $enF^{1/s} \leq \lambda_{\max} = \text{poly}(n)$*. Let* $\kappa := \frac{\log \log n}{\log \log \log n}$*. For every initial* $\lambda_0 \leq \lambda_{\max}$ *and every initial search point* x_0 *with* $|x_0|_1 \geq 2n/3$ *the algorithm reaches a* κ*-safe state, or a search point with at least* $3n/4$ *ones, in* $O(\log(\lambda_{\max}) \log n)$ *expected generations and* $O(\lambda_{\max} \log n)$ *expected evaluations.*

The main idea of the proof of Lemma 4.5 is that, once the algorithm reaches a local optimum with $2n/3$ ones, λ will increase until it resets to 1 and this is repeated until the algorithm leaves its local optimum. Every time $\lambda = 1$ the algorithm will accept any offspring, then we can wait until a lucky mutation step can directly jump to a search point with at least $2n/3 + \frac{\log \log n}{\log \log \log n}$ ones. Finally, we account for the time that the algorithm takes to reset λ to 1, including the time spent outside of local optima in states that are not safe.

One such set of non-safe states is that of states with at least $2n/3 + \kappa$ ones but violating the upper bound for λ_t from Definition 4.4. For these states we cannot exclude that the algorithm will return to the first slope before it reaches a safe state, but we can bound the time the algorithm spends before either event happens and later on account for this time. This is done in the following lemma.

Lemma 4.6. *Consider the self-adjusting $(1, \lambda)$ EA with resetting λ as in Theorem 4.1. Let $\kappa := \frac{\log\log n}{\log\log\log n}$. From any state (x_0, λ_0) with $|x_0|_1 \geq 2n/3 + \kappa$ in expectation the algorithm needs at most $O(\log \lambda_{\max})$ generations and $O(\lambda_{\max})$ evaluations to reach a κ-safe state, to return to the first slope or to find a search point x_t with $|x_t|_1 \geq 3n/4$.*

The main proof idea is to show that with a large value of λ, with high probability all generations are successful and λ is quickly reduced to a safe value, unless any of the other two events happens.

Proof of Lemma 4.6. In every successful generation the algorithm will decrease λ and either increase the number of ones or go back to the first slope. In the latter case, we are done. In the first case, the algorithm moves towards a κ-safe state as λ_t is decreased. If the algorithm only has successful generations it will reach a κ-safe state once λ has decreased below the threshold of $2^{-\kappa/2} \cdot (\kappa/2)!$. Assuming that every generation is successful, after i generations the algorithm will reduce the offspring population size to $\lambda = \frac{\lambda_0}{F^i}$. For $i = \alpha := \left\lceil \log_F \left(\frac{\lambda_0}{2^{-\kappa/2}\cdot(\kappa/2)!} \right) \right\rceil$, we get an offspring population size of

$$\frac{\lambda_0}{F^{\left\lceil \log_F \left(\frac{\lambda_0}{2^{-\kappa/2}\cdot(\kappa/2)!} \right) \right\rceil}} \leq \frac{\lambda_0}{F^{\log_F \left(\frac{\lambda_0}{2^{-\kappa/2}\cdot(\kappa/2)!} \right)}} = 2^{-\kappa/2} \cdot (\kappa/2)!.$$

Hence the algorithm needs at most α consecutive successful generations to obtain $\lambda \leq 2^{-\kappa/2} \cdot (\kappa/2)!$. During these generations, the number of evaluations is at most

$$\begin{aligned}\sum_{i=0}^{\alpha-1} \left\lceil \frac{\lambda_0}{F^i} \right\rceil &\leq \sum_{i=0}^{\alpha-1} \left(\frac{\lambda_0}{F^i} + 1 \right) \leq \alpha + \lambda_0 \sum_{i=0}^{\infty} \left(\frac{1}{F} \right)^i \\ &= \alpha + \lambda_0 \left(\frac{1}{1 - 1/F} \right) = \alpha + \lambda_0 \left(\frac{F}{F-1} \right) \\ &= O(\lambda_0) = O(\lambda_{\max}).\end{aligned}$$

We now show that the algorithm finds an improvement in every generation with probability $1 - o(1)$. The algorithm will increase λ in all generations that are unsuccessful. Since we only consider $|x_t|_1 \leq 3n/4$ the probability of an offspring being better than the parent is at least $1/(4e)$ and the probability of an unsuccessful generation is at most $\left(1 - \frac{1}{4e}\right)^{\lambda}$. By the union bound the probability that there is at least one unsuccessful generation during α generations is at most

$$\sum_{j=0}^{\alpha-1} \left(1 - \frac{1}{4e}\right)^{\frac{\lambda_0}{F^j}} \leq \sum_{j=0}^{\alpha-1} e^{-\frac{\lambda_0}{4eF^j}}.$$

We now show by induction that for every $\alpha \geq 1$ the inequality $\sum_{j=0}^{\alpha-1} e^{-\frac{\lambda_0}{4eF^j}} \leq 2e^{-\frac{\lambda_0}{4eF^{\alpha-1}}}$ holds. Starting with the base case $\alpha = 1$ the statement is clearly true:

$$\sum_{j=0}^{0} e^{-\frac{\lambda_0}{4eF^j}} = e^{-\frac{\lambda_0}{4e}} \leq 2e^{-\frac{\lambda_0}{4e}}.$$

We now show that if the statement holds for $\alpha - 1$ then it also holds for α. First we note that, since $2^{-\kappa/2} \cdot (\kappa/2)! = \omega(1)$, the following holds for large enough n:

$$\begin{aligned}2 &\leq \exp\left(\frac{F-1}{4e} \cdot \frac{2^{-\kappa/2} \cdot (\kappa/2)!}{F} \right) \\ &= \exp\left(\frac{F-1}{4e} \cdot \frac{\lambda_0}{F^{\log_F \left(\frac{\lambda_0}{2^{-\kappa/2}\cdot(\kappa/2)!} \right) + 1}} \right) \\ &\leq \exp\left(\frac{F-1}{4e} \cdot \frac{\lambda_0}{F^{\left\lceil \log_F \left(\frac{\lambda_0}{2^{-\kappa/2}\cdot(\kappa/2)!} \right) \right\rceil}} \right) = \exp\left(\frac{F-1}{4e} \cdot \frac{\lambda_0}{F^{\alpha}} \right)\end{aligned}$$

Hence, using the inductive hypothesis the following statement holds:

$$\sum_{j=0}^{\alpha-1} e^{-\frac{\lambda_0}{4eF^j}} \leq 2e^{-\frac{\lambda_0}{4eF^{\alpha-1}}} \leq e^{\frac{(F-1)\lambda_0}{4eF^{\alpha}}} \cdot e^{-\frac{\lambda_0}{4eF^{\alpha-1}}} = e^{\left(\frac{(F-1)\lambda_0}{4eF^{\alpha}} - \frac{\lambda_0}{4eF^{\alpha-1}} \right)} = e^{\left(\frac{(F-1)\lambda_0 - F\lambda_0}{4eF^{\alpha}} \right)} = e^{-\frac{\lambda_0}{4eF^{\alpha}}}.$$

Now adding $e^{-\frac{\lambda_0}{4eF^{\alpha}}}$ to the first and last term we obtain

$$\sum_{j=0}^{\alpha} e^{-\frac{\lambda_0}{4eF^j}} \leq 2e^{-\frac{\lambda_0}{4eF^\alpha}},$$

proving the hypothesis.

Therefore, the probability that there are α consecutive successful generations yielding $\lambda \leq 2^{-\kappa/2} \cdot (\kappa/2)!$ is at least

$$1 - 2e^{-\frac{\lambda_0}{4eF^{\alpha-1}}} \geq 1 - 2e^{-\frac{\lambda_0}{4e\left(\frac{\lambda_0}{2^{-\kappa/2}\cdot(\kappa/2)!}\right)}} = 1 - 2e^{-\frac{2^{-\kappa/2}\cdot(\kappa/2)!}{4e}} = 1 - o(1).$$

With the same probability the algorithm will reduce λ to a value $\lambda_t \leq 2^{-\kappa/2} \cdot (\kappa/2)!$ in

$$\left\lceil \log_F \left(\frac{\lambda_0}{2^{-\kappa/2} \cdot (\kappa/2)!} \right) \right\rceil = O(\log(\lambda_0)) = O(\log \lambda_{\max})$$

generations and $O(\lambda_{\max})$ evaluations. If there is an unsuccessful generation we can restart the arguments. Since there are no unsuccessful generations with probability $1 - o(1)$, we need to restart the arguments at most $1 + o(1)$ times and obtain the same asymptotic expected number of generations and evaluations. □

There is one last missing event that we need to take into account: The algorithm may "slip down" the first slope from a local optimum. To account for this event we use a result shown in our previous work [26] where we show that the algorithm does not decrease the number of one-bits by more than $O(\log n)$ with high probability. Since we use this result again in later proofs, we state the lemma here for completeness.

Lemma 4.7 *(Lemma 3.7 in [26]). Consider the self-adjusting* $(1, \lambda)$ *EA on* ONEMAX *without resets and with* $f(x_t) := \text{ONEMAX}(x_t)$*. Let the update strength* $F > 1$ *and the success rate* $0 < s < 1$ *be constants. 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:* $f(x_t) \geq f_t^* - r \log n$*.*

Note that the lemma is stated for ONEMAX instead of CLIFF, and it considers the self-adjusting $(1, \lambda)$ EA without resets. We note that the condition $\lambda_t \geq 4 \log n$ can only happen if $\lambda_{\max} \geq 4 \log n$. This is already taken into account by Theorem 4.1 since the condition $\lambda_{\max} \geq enF^{1/s}$ already covers $\lambda_{\max} \geq 4 \log n$ as $enF^{1/s} \geq 4 \log n$ for all $F > 1$ $s > 0$ and $n > 0$.

The lemma was proven using drift analysis, and the drift towards increased numbers of ones was estimated by only considering steps increasing the number of ones by exactly 1. This means that the statements can be directly transferred to CLIFF when considering search points with less than $2n/3$ ones. (We remark for later use that the lemma also applies to search points on the second slope, so long as the algorithm does not go back to the first slope.) With Lemmas 4.6 and 4.7 we are able to prove Lemma 4.5.

Proof of Lemma 4.5. We define a cycle as a sequence of generations that begins after λ is reset to 1 and $|x_t|_1 = 2n/3$. Since $|x_0|_1 \geq 2n/3$ and there is no assumption on λ_0 we need to take into a account the time taken to start the first cycle; this will be accounted later on.

By Lemma 2.4 with $c := 1$ in the first generation of every cycle (with $\lambda = 1$) there is a probability of at least $1/\log n$ to create and accept an offspring with at least $2n/3 + \frac{\log\log n}{\log\log\log n}$ ones. The next generation will have $\lambda = F^{1/s}$, which for a sufficiently large n satisfies $\lambda \leq 2^{-\kappa/2} \cdot (\kappa/2)!$. Hence, in expected $\log n$ cycles the sought event will happen. Now it remains to bound the expected number of generations and evaluations in each cycle.

If during a cycle all generations maintain the current fitness value, after j generations the offspring population size is $F^{j/s}$. For $j := \lceil s \log_F \lambda_{\max} \rceil$, we get an offspring population size of

$$F^{\lceil s \log_F \lambda_{\max} \rceil / s} \geq F^{\log_F \lambda_{\max}} = \lambda_{\max}.$$

Thus, the number of generations needed to reset λ is at most $\lceil s \log_F \lambda_{\max} \rceil + 1$. Using $\lceil F^{j/s} \rceil \leq 2F^{j/s}$, during these generations, the number of evaluations is at most

$$\begin{aligned}
\sum_{j=0}^{\lceil s \log_F \lambda_{\max} \rceil} \left\lceil F^{j/s} \right\rceil &\leq 2 \sum_{j=0}^{\lceil s \log_F \lambda_{\max} \rceil} \left(F^{1/s}\right)^j \\
&= 2 \cdot \frac{(F^{1/s})^{\lceil s \log_F(\lambda_{\max}) \rceil + 1} - 1}{F^{1/s} - 1} \\
&\leq 2 \cdot \frac{(F^{1/s})^{s \log_F(\lambda_{\max}) + 2}}{F^{1/s} - 1}
\end{aligned}$$

$$= \frac{2F^{2/s}}{F^{1/s}-1} \cdot \lambda_{\max} = O(\lambda_{\max}).$$

We now show that when the algorithm is in a local optimum, with constant probability all following generations will maintain the current fitness value until λ is reset to 1. When $|x_t|_1 = 2n/3$, in order for a generation to maintain the number of one-bits, it is sufficient to create at least one copy of the parent. Hence, the probability of the event is at least

$$1-\left(1-\left(1-\frac{1}{n}\right)^n\right)^\lambda \geq \frac{\lambda\left(1-\frac{1}{n}\right)^n}{1+\lambda\left(1-\frac{1}{n}\right)^n} = \frac{1}{1+\frac{1}{\lambda\left(1-\frac{1}{n}\right)^n}} \geq \frac{1}{\exp\left(\frac{1}{\lambda\left(1-\frac{1}{n}\right)^n}\right)} = \exp\left(-\frac{1}{\lambda\left(1-\frac{1}{n}\right)^n}\right).$$

The probability that a cycle is comprised only of generations that maintain the fitness value is at least

$$\prod_{j=0}^{\lceil s\log_F \lambda_{\max}\rceil} \exp\left(-\frac{1}{F^j\left(1-\frac{1}{n}\right)^n}\right) \geq \prod_{j=0}^{\infty} \exp\left(-\frac{1}{F^j\left(1-\frac{1}{n}\right)^n}\right) = \exp\left(-\frac{1}{\left(1-\frac{1}{n}\right)^n}\sum_{j=0}^{\infty} F^{-j}\right) = \exp\left(-\frac{1}{\left(1-\frac{1}{n}\right)^n}\cdot\frac{F}{F-1}\right) = \Omega(1).$$

Therefore, in each cycle the algorithm will directly increase λ to $\lambda_{\max}$ and then reset to 1 with constant probability using $O(\log\lambda_{\max})$ generations and $O(\lambda_{\max})$ evaluations.

Since the self-adjusting $(1,\lambda)$ EA is non-elitist, there is still a possibility for the algorithm to either jump down the cliff (but not to the desired κ-safe state) or to reduce the number of ones. We consider the total time T_c until a cycle finishes, that is, λ resets to 1 with the current search point having $2n/3$ ones. We use as superscript gen and eval to denote that we are considering number of generations or evaluations, respectively. Similar to the proof of Theorem 3.1, let p^- denote the probability of accepting a search point with less than $2n/3$ ones from the top of the cliff and let p^+ denote the probability of accepting a search point with more than $2n/3$ ones from the top of the cliff.

Now, let T_1^{gen} denote the number of generations and T_1^{eval} denote the number of evaluations to return to the top of the cliff after accepting a search point with less than $2n/3$ ones. If the number of one-bits is reduced, assuming no reset occurs, by Lemma 4.7 with probability $1-O(1/n)$ the number of one-bits will never drop below $2n/3 - O(\log n)$ ones before reaching a point with $2n/3$ ones. Note that during this time by Lemma 4.2 with overwhelming probability no reset occurs. The overwhelmingly small error probability can be easily absorbed in the $-O(1/n)$ term from the above probability bound $1-O(1/n)$. By Lemma 4.3, it will take $O(\log n)$ generations and $O(\log n + \lambda_0) = O(\lambda_{\max})$ evaluations in expectation to return to a local optimum. With probability $O(1/n)$ the number of one-bits will drop below $2n/3 - O(\log n)$ and again by Lemma 4.3 in expectation it will take at most $O(n)$ generations and $O(n+\lambda_0) = O(\lambda_{\max})$ to return to a local optimum. Hence,

$$\mathrm{E}\left(T_1^{\text{gen}}\right) = (1-O(1/n))O(\log n) + O(1/n)O(n) = O(\log n),$$
$$\mathrm{E}\left(T_1^{\text{eval}}\right) = (1-O(1/n))O(\lambda_{\max}) + O(1/n)O(n) = O(\lambda_{\max}).$$

Let T_2^{gen} and T_2^{eval} denote the number of generations and evaluations to return to the first slope or finding κ-safe state from a search point on the second slope that is not in a κ-safe state. Using the same arguments as in Lemma 4.3 we can see that the algorithm will use $O(\log n)$ generations and $O(\lambda_{\max})$ evaluations to either find a solution with at least $2n/3 + \frac{\log\log n}{\log\log\log n}$ ones or jump back to the first slope. If the algorithm finds a solution with at least $2n/3 + \frac{\log\log n}{\log\log\log n}$ ones but with $\lambda > 2^{-\kappa/2}\cdot(\kappa/2)!$ then by Lemma 4.6 it will take $O(\log\lambda_{\max})$ generations and $O(\lambda_{\max})$ evaluations in expectation to either reduce λ to $\lambda \leq 2^{-\kappa/2}\cdot(\kappa/2)!$ or to return to the first slope. Hence, we obtain

$$\mathrm{E}\left(T_2^{\text{gen}}\right) = O(\log n) + O(\log\lambda_{\max}) = O(\log\lambda_{\max}),$$
$$\mathrm{E}\left(T_2^{\text{eval}}\right) = O(\lambda_{\max}) + O(\lambda_{\max}) = O(\lambda_{\max}).$$

Finally, when jumping back to the first slope, we will reach a search point x with $|x|_1 - a$ and $a \geq 0$ is a random variable. By Lemma 2.3 we have $\mathrm{E}(a) \leq 1$, that is, in expectation the jump overshoots the top of the cliff by a distance of at most 1. Let T_3^{eval} be the time to go back to $|x|_1 = 2n/3$ from $|x|_1 = 2n/3 - a$. By the law of total expectation and Lemma 4.3, $\mathrm{E}\left(T_3^{\text{eval}}\right) = \mathrm{E}(\mathrm{E}(T \mid a)) \leq \mathrm{E}\left(O(a + \log n + \lambda_0)\right) = O(\mathrm{E}(a)) + O(\log n + \lambda_0)$. Given that $\mathrm{E}(a) \leq 1$ we obtain $\mathrm{E}\left(T_3^{\text{eval}}\right) = O(\log n + \lambda_0) = O(\lambda_{\max})$ evaluations. For the number of generations T_3^{gen} we use the same arguments and obtain $O(\log n)$ generations.

Therefore, if the algorithm moves out of the local optimum it will return to it in

$$O(\log\lambda_{\max})\Omega(1) + p^-\mathrm{E}\left(T_1^{\text{gen}}\right) + p^+(\mathrm{E}\left(T_2^{\text{gen}}\right) + \mathrm{E}\left(T_3^{\text{gen}}\right)) = O(\log\lambda_{\max})$$

expected generations and

$$O(\lambda_{\max})\Omega(1) + p^-\mathrm{E}\left(T_1^{\text{eval}}\right) + p^+(\mathrm{E}\left(T_2^{\text{eval}}\right) + \mathrm{E}\left(T_3^{\text{eval}}\right)) = O(\lambda_{\max})$$

expected evaluations.

It remains to account for the time before the first cycle. Using the same arguments as before for any $\lambda_0 \le \lambda_{\max}$ and $|x_0|_1 \ge 2n/3$ the algorithm will spend $O(\log \lambda_{\max})$ generations and $O(\lambda_{\max})$ evaluations to start the first cycle or reach the desired state. Noting that the expected number of cycles is $\log n$ proves the claim. $\square$

4.3. After jumping down the cliff

Now we show that, with probability $\Omega(1)$, we reach a search point with at least $3n/4$ ones when starting from a κ-safe state with $\kappa := \frac{\log\log n}{\log\log\log n}$. As in Section 3, the target of reaching $3n/4$ ones is chosen such that the probability of an improving mutation is always $\Omega(1)$.

Proving this claim is not straightforward for several reasons. It is always possible to have a mutation jumping back up the cliff. This probability decreases with the distance to the cliff (that is, $|x_t|_1 - 2n/3$) and it increases with λ_t as many offspring can amplify the probability of a jump back up the cliff (cf. Lemma 2.3). Fortunately, the notion of κ-safe states implies that we start with a distance of at least κ to the cliff and a small λ_t, so that initially this probability amplification does not pose a huge risk.

But small values of λ are risky for another reason. Since the number of ones is larger than $2n/3$ and hence significantly larger than $n/2$, the expected number of ones in any offspring is smaller than that of its parent. With $\lambda_t \approx 1$ there is a constant negative drift towards decreasing the number of zeros and "slipping down" the second slope. Fortunately, λ_t will increase during unsuccessful generations and we will see that this effect prevents the algorithm from slipping down the second slope.

In order to analyse the self-adjusting $(1, \lambda)$ EA on OneMax, in our earlier work [26] we used a potential function such that it differs from the fitness by an additive term of $O(\log n)$. This characteristic of the potential function was critical to construct the so-called "ratchet argument", where we showed that with high probability, the best fitness never decreases by a term of $r \log n$, for some constant $r > 0$.

Unfortunately, this ratchet argument is not directly applicable here, since we can only guarantee a distance of $\kappa := \frac{\log\log n}{\log\log\log n} \ll r \log n$ to the cliff. Hence the ratchet argument from [26] has far too much slack.

The proof of the ratchet argument in [26] applies the negative drift theorem [40,41] to an interval on the potential scale of size $\Theta(\log n)$, in order to obtain failure probabilities that are polynomially small (that is, exponentially small in the interval length).

We refine the ratchet argument here by defining a revised potential function tailored to a fitness range up to $3n/4$ ones, where the fitness and the potential only differ by an additive term of $\Theta(1)$. Then we apply the negative drift theorem [40,41] to an interval of size $\kappa/2 - O(1) = \frac{\log\log n}{2\log\log\log n} - O(1)$ on the potential scale to show that the number of ones does not drop below $2n/3 + \kappa/2$ in a time that is exponential in the interval length. More specifically, the time period will be determined as γ^κ, for some constant $\gamma > 1$. During this time, the potential has a positive drift and with good probability the algorithm moves sufficiently far away from the cliff, that is, to a distance of $\Theta(\gamma^\kappa)$.

Since $\gamma^{\frac{\log\log n}{\log\log\log n}} = o(\log n)$, we can only guarantee a sub-logarithmic increase in the distance and the failure probability from the negative drift theorem is $\omega(1/\log n)$. Thus, we iterate this argument two more times, with exponentially increasing values for κ, until we reach a search point with at least $3n/4$ ones (or we return to the first slope).

Throughout these arguments, we also show that λ_t is bounded from above as $\lambda_t \le 2^{-\kappa/2} \cdot (\kappa/2)!$ as in the definition of κ-safe states. This definition requires a distance of at least κ from the top of the cliff, however we can only guarantee a distance of at least $\kappa/2$. We call such states *weakly κ-safe*.

Definition 4.8. A state (x_t, λ_t) is called *weakly κ-safe* if $|x_t|_1 \ge 2n/3 + \kappa/2$ and $\lambda_t \le 2^{-\kappa/2} \cdot (\kappa/2)!$.

We start by revising the potential function from [26][2] as follows.

Definition 4.9. Let $\varepsilon := \left(\frac{e-1}{e}\right)\left(\frac{1-s}{s+1}\right)$. We define the potential function $g(X_t)$ as

$$g(X_t) = |x_t|_1 - \frac{2s}{s+1} \log_F \left(\max \left(\frac{F^{1/s}(8e + \log_{\frac{e}{e-1}}(2/\varepsilon))}{\lambda_t}, 1 \right) \right).$$

This potential function contains two distinct terms. The first term takes into account the progress towards the optimum by considering the number of one-bits (we call it *number-of-ones component*) and the second term takes into account how λ changes over time. In particular we use the second term as a *penalty* for small λ-values that reduces when λ increase and disappears completely when λ is sufficiently large for the algorithm to progress towards the optimum. Note that the argument of the logarithm is in $\Theta(1)$ and the penalty term is in $-O(1)$ and at most 0.

The potential function allows us to show a positive drift in the potential for all the fitness levels considered and all λ because in the generations where the fitness decreases, the penalty is reduced.

[2] Similar approaches have been used before for analysing self-adjusting mutation rates [42,24] and for continuous domains in [43–45].

Lemma 4.10. *Consider the self-adjusting* $(1, \lambda)$ *EA as in Theorem 4.1. For all states* (x_t, λ_t) *with* $d := |x_t|_1 - 2n/3 = \omega(1)$, $|x_t|_1 \le 3/4$ *and* $\lambda_t \le 2^{-d/2} \cdot (d/2)!$,

$$E\left(g(X_{t+1}) - g(X_t) \mid X_t\right) \ge \frac{1-s}{4e} > 0$$

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

Proof. The proof follows the proof of Lemma 3.4 in [26], using additional arguments to consider jumps up the cliff and the possibility that λ is reset when $\lambda_t = \lambda_{\max}$. In this proof, we use $p^0_{i,\lambda} := 1 - p^{\rightarrow}_{i,\lambda} - p^{\leftarrow}_{i,\lambda} - p^{\uparrow}_{i,\lambda}$ to denote the probability of not changing the current number of ones. We first assume that $\lambda_{\max} > 2^{-d/2} \cdot (d/2)!$, which implies that resets are impossible under the assumption $\lambda_t \le 2^{-d/2} \cdot (d/2)!$.

We first consider the case $\lambda_t \le 8e + \log_{\frac{e}{e-1}}(2/\varepsilon)$ as then $\lambda_{t+1} \le F^{1/s}(8e + \log_{\frac{e}{e-1}}(2/\varepsilon))$ and $g(X_{t+1}) = |x_{t+1}|_1 - \frac{2s}{s+1}(\log_F(F^{1/s}(8e + \log_{\frac{e}{e-1}}(2/\varepsilon))) - \log_F(\lambda_{t+1}))$. When the number of ones increases, in expectation they do so by $\Delta^{\rightarrow}_{i,\lambda}$ and since $\lambda_{t+1} = \lambda_t/F$, the penalty term $\frac{2s}{s+1}(\log_F(F^{1/s}(8e + \log_{\frac{e}{e-1}}(2/\varepsilon))) - \log_F(\lambda_t))$ increases by $\frac{2s}{s+1}$ (unless $\lambda_{t+1} = 1$ is reached, in which case the increase might be lower). When the number of ones does not change, the penalty decreases by $\frac{2}{s+1}$. When the number of ones decreases, conditional on $|x_{t+1}|_1 > 2n/3$, the expected decrease is at most $\Delta^{\leftarrow}_{i,\lambda}$ and the penalty decreases by $\frac{2}{s+1}$. Finally, then when the algorithm creates an offspring up the cliff the expected decrease in the number of ones is $\Delta^{\uparrow}_{i,\lambda}$ and the penalty increases by $\frac{2s}{s+1}$. Together,

$$\begin{aligned}
E\left(g(X_{t+1}) - g(X_t) \mid X_t \cap \lambda_t \le 8e + \log_{\frac{e}{e-1}}(2/\varepsilon)\right) &\ge p^{\rightarrow}_{i,\lambda}\left(\Delta^{\rightarrow}_{i,\lambda} - \frac{2s}{s+1}\right) + \frac{2p^0_{i,\lambda}}{s+1} + p^{\leftarrow}_{i,\lambda}\left(\frac{2}{s+1} - \Delta^{\leftarrow}_{i,\lambda}\right) + p^{\uparrow}_{i,\lambda}\left(-\Delta^{\uparrow}_{i,\lambda} - \frac{2s}{s+1}\right) \\
&= p^{\rightarrow}_{i,\lambda}\left(\Delta^{\rightarrow}_{i,\lambda} - \frac{2s}{s+1}\right) + \frac{2(p^0_{i,\lambda} + p^{\leftarrow}_{i,\lambda})}{s+1} - \Delta^{\leftarrow}_{i,\lambda}p^{\leftarrow}_{i,\lambda} - p^{\uparrow}_{i,\lambda}\left(\Delta^{\uparrow}_{i,\lambda} + \frac{2s}{s+1}\right) \\
&= p^{\rightarrow}_{i,\lambda}\left(\Delta^{\rightarrow}_{i,\lambda} - \frac{2s}{s+1}\right) + \frac{2(1 - p^{\rightarrow}_{i,\lambda} - p^{\uparrow}_{i,\lambda})}{s+1} - \Delta^{\leftarrow}_{i,\lambda}p^{\leftarrow}_{i,\lambda} - p^{\uparrow}_{i,\lambda}\left(\Delta^{\uparrow}_{i,\lambda} + \frac{2s}{s+1}\right) \\
&= p^{\rightarrow}_{i,\lambda}\left(\Delta^{\rightarrow}_{i,\lambda} - 2\right) + \frac{2}{s+1} - \Delta^{\leftarrow}_{i,\lambda}p^{\leftarrow}_{i,\lambda} - p^{\uparrow}_{i,\lambda}\left(\Delta^{\uparrow}_{i,\lambda} + 2\right).
\end{aligned}$$

Using $\Delta^{\rightarrow}_{i,\lambda} \ge 1$ (which also holds when only considering fitness increases by 1), and $\Delta^{\uparrow}_{i,\lambda} \le d+1$ by Lemma 2.3, this is at least

$$1 + \frac{1-s}{s+1} - p^{\rightarrow}_{i,\lambda} + p^{\leftarrow}_{i,\lambda}\Delta^{\leftarrow}_{i,\lambda} - p^{\uparrow}_{i,\lambda}(d+3).$$

Following the calculations from Lemma 3.4 in [26] $1 + \frac{1-s}{s+1} - p^{\rightarrow}_{i,\lambda} + p^{\leftarrow}_{i,\lambda}\Delta^{\leftarrow}_{i,\lambda}$ is at least $\frac{1-s}{2e}$. By Lemma 2.3 $p^{\uparrow}_{i,\lambda} \le \frac{\lambda(3/4)^d}{d!}$, in addition by assumption $d = \omega(1)$ and $\lambda \le 8e + \log_{\frac{e}{e-1}}(2/\varepsilon) = O(1)$. Therefore, $p^{\uparrow}_{i,\lambda}(d+3) = o(1)$ and for a sufficiently large n,

$$E\left(g(X_{t+1}) - g(X_t) \mid X_t \cap \lambda_t \le 8e + \log_{\frac{e}{e-1}}(2/\varepsilon)\right) \ge \frac{1-s}{2e} - o(1) \ge \frac{1-s}{4e}.$$

For the case $8e + \log_{\frac{e}{e-1}}(2/\varepsilon) < \lambda_t \le 2^{-d/2} \cdot (d/2)!$, 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 increasing the number of ones is large enough to show a positive drift.

By assumption $\lambda_t \le 2^{-d/2} \cdot (d/2)!$. Along with (5) from Lemma 2.3,

$$p^{\uparrow}_{i,\lambda} \le \frac{\lambda(3/4)^d}{d!} \le \left(\frac{3}{8}\right)^d.$$

We also have $\lambda_t > 8e + \log_{\frac{e}{e-1}}(2/\varepsilon) > 8e$ since $1/\varepsilon \ge \frac{e}{e-1}$. Then, by Lemma 2.3, $8e + \log_{\frac{e}{e-1}}(2/\varepsilon) < \lambda_t \le 2^{-d/2} \cdot (d/2)!$ implies the following two statements.

$$\begin{aligned}
p^{\rightarrow}_{i,\lambda} &\ge 1 - \left(1 - \frac{1}{4e}\right)^{8e} - \left(\frac{3}{8}\right)^d \ge 1 - \frac{1}{2e} - \left(\frac{3}{8}\right)^d \\
p^{\leftarrow}_{i,\lambda}\Delta^{\leftarrow}_{i,\lambda} &\le \left(\frac{e-1}{e}\right)^{8e+\log_{\frac{e}{e-1}}(2/\varepsilon)} \cdot \frac{e}{e-1} = \left(\frac{e-1}{e}\right)^{8e-1+\log_{\frac{e}{e-1}}(2/\varepsilon)} \le \left(\frac{e-1}{e}\right)^{\log_{\frac{e}{e-1}}(2/\varepsilon)} = \frac{\varepsilon}{2}.
\end{aligned}$$

Together,

$$\begin{aligned}
&E\left(g(X_{t+1}) - g(X_t) \mid X_t \cap 8e + \log_{\frac{e}{e-1}}(2/\varepsilon) < \lambda_t < 2^{-d/2} \cdot (d/2)!\right) \\
&\ge p^{\rightarrow}_{i,\lambda}\left(1 - \frac{2s}{s+1}\right) + p^{\leftarrow}_{i,\lambda}\left(-\Delta^{\leftarrow}_{i,\lambda}\right) - p^{\uparrow}_{i,\lambda}\left(d + 1 + \frac{2s}{s+1}\right) \\
&\ge \left(1 - \frac{1}{2e} - \left(\frac{3}{8}\right)^d\right)\left(1 - \frac{2s}{s+1}\right) - \frac{\varepsilon}{2} - \left(\frac{3}{8}\right)^d(d+2)
\end{aligned}$$

given that $d = \omega(1)$,

$$\geq \left(1-\frac{1}{e}\right)\left(1-\frac{2s}{s+1}\right)-\frac{\varepsilon}{2}-o(1)=\left(\frac{e-1}{e}\right)\left(\frac{1-s}{s+1}\right)-\frac{\varepsilon}{2}-o(1)$$

using the definition $\varepsilon := \left(\frac{e-1}{e}\right)\left(\frac{1-s}{s+1}\right)$,

$$=\left(\frac{e-1}{2e}\right)\left(\frac{1-s}{s+1}\right)-o(1).$$

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

$$\mathrm{E}\left(g(X_{t+1})-g(X_t)\mid X_t\cap\lambda_t\leq 8e+\log_{\frac{e}{e-1}}(2/\varepsilon)\right)\geq\frac{1}{2e}\left(\frac{1-s}{s+1}\right)\geq\frac{1-s}{4e}.$$

Since $s<1$, this is a strictly positive constant.

Now, if $\lambda_{\max}\leq 2^{-d/2}\cdot(d/2)!$ then resets may happen if $\lambda_t=\lambda_{\max}$. A reset decreases the potential by at most $n+O(1)$ as this is the range of the potential scale. The probability of a reset is at most $e^{-\Omega(n)}$ by Lemma 4.2. Hence, this only affects the drift by an additive term $-O(n)\cdot e^{-\Omega(n)}=-o(1)$, which can easily be absorbed in the $-o(1)$ terms from the above calculations. □

The following lemma shows that λ typically does not grow beyond the threshold $2^{-\kappa/2}\cdot(\kappa/2)!$ from the definition of weakly κ-safe states.

Lemma 4.11. *Consider the self-adjusting $(1,\lambda)$ EA as in Theorem 4.1. Then for all $\kappa\geq 324F^{1/s}$ the following holds. If the current state (x_t,λ_t) has $\lambda_t\leq 2^{-\kappa/2}\cdot(\kappa/2)!$ and $2n/3<|x_t|_1<3n/4$, then with probability at least $1-2^{-2\kappa}$ we have $\lambda_{t+1}\leq 2^{-\kappa/2}\cdot(\kappa/2)!$.*

Proof. Since $\lambda_t\leq 2^{-\kappa/2}\cdot(\kappa/2)!$, a necessary condition for $\lambda_{t+1}>2^{-\kappa/2}\cdot(\kappa/2)!$ is that generation t is unsuccessful. Since $|x_t|_1<3n/4$, the probability of finding an improvement in any mutation is at least $1/(4e)$ and the probability of an unsuccessful generation is at most

$$\left(1-\frac{1}{4e}\right)^{F^{-1/s}2^{-\kappa/2}(\kappa/2)!}\leq\left(\frac{4e}{4e-1}\right)^{-F^{-1/s}(2e)^{-\kappa/2}(\kappa/2)^{\kappa/2}}=2^{-F^{-1/s}(\kappa/(4e))^{\kappa/2}\log\left(\frac{4e}{4e-1}\right)}. \tag{16}$$

The condition $\kappa\geq 324F^{1/s}$ implies

$$\frac{\kappa}{4e}\geq\left(1+\frac{4}{\log\left(\frac{4e}{4e-1}\right)}\right)F^{1/s}\geq\left(1+\frac{4F^{1/s}}{\log\left(\frac{4e}{4e-1}\right)}\right).$$

We bound the absolute value of the exponent in (16) using $(1+y)^x\geq xy$ for $x\in\mathbb{N}_0$, $y\geq 0$, as follows.

$$\begin{aligned}F^{-1/s}(\kappa/(4e))^{\kappa/2}\log\left(\frac{4e}{4e-1}\right)&\geq F^{-1/s}\left(1+\frac{4F^{1/s}}{\log\left(\frac{4e}{4e-1}\right)}\right)^{\kappa/2}\log\left(\frac{4e}{4e-1}\right)\\&\geq F^{-1/s}\frac{4F^{1/s}}{\log\left(\frac{4e}{4e-1}\right)}\cdot\kappa/2\cdot\log\left(\frac{4e}{4e-1}\right)=2\kappa.\end{aligned}$$

Hence the probability of an unsuccessful generation is at most $2^{-2\kappa}$. □

The following lemma now generalises and refines the "ratchet argument" from [26].

Lemma 4.12. *Consider the self-adjusting $(1,\lambda)$ EA as in Theorem 4.1. Let $T_{3n/4}=\inf\{t\mid |x_t|_1\geq 3n/4\}$ be the number of generations until a search point with at least $3n/4$ ones is reached.*

There are constants $\gamma:=\gamma(s,F)\in(1,2]$ and $\kappa_0:=\kappa_0(s,F,\gamma)\geq 2$ such that for all $\kappa\geq\kappa_0$ the following holds. If the initial state (x_0,λ_0) is κ-safe with $|x_0|_1<3n/4$ then with probability at least $1-\gamma^{-\Omega(\kappa)}$ all states during the next $\min\{\gamma^\kappa,T_{3n/4}\}$ generations are weakly κ-safe.

Proof. Let (x_0,λ_0) denote the initial state of the self-adjusting $(1,\lambda)$ EA. If $|x_0|_1\geq 3n/4$ the statement is trivial, hence we assume $|x_0|_1<3n/4$. As in the proof of Lemma 3.7 in [26], we are setting up to apply the negative drift theorem [40,41].

The value of γ will be determined later on, ensuring $1<\gamma\leq 2$. Choosing $\kappa_0\geq 324F^{1/s}$ and recalling that the initial state is κ-safe, Lemma 4.11 states that, with probability at least $1-2^{-2\kappa}$, $\lambda_1\leq 2^{-\kappa/2}\cdot(\kappa/2)!$ as long as the number of ones is smaller than $3n/4$. By induction and a union bound, this holds for the first $\min\{\gamma^\kappa,T_{3n/4}\}$ generations with probability at least $1-2^{-2\kappa}\cdot\min\{\gamma^\kappa,T_{3n/4}\}\geq 1-2^{-2\kappa}\cdot\gamma^\kappa\geq 1-\gamma^{-k}$ as $\gamma\leq 2$, unless the algorithm jumps back to the first slope. We assume in the following that the bound on λ_t always applies, while the algorithm remains on the second slope (and has not found a search point with at least $3n/4$ ones yet).

Let $\alpha := \frac{se}{e-1} \log_F \left(8e + \log_{\frac{e}{e-1}} (2/\varepsilon) F^{1/s} \right) = O(1)$ abbreviate the maximum difference between the potential g and the number of ones, then we start with a potential of at least $2n/3 + \kappa - \alpha$. We apply the negative drift theorem [40,41] to an interval $[a, b] := [2n/3 + \kappa/2, 2n/3 + \kappa - \alpha]$ with respect to the current potential. By choosing $\kappa_0 \geq 6\alpha$, we can ensure that $b - a = \kappa - \alpha - \kappa/2 = \kappa/3 + \kappa/6 - \alpha \geq \kappa/3$.

We pessimistically assume that the number-of-ones component of g can only increase by at most 1. Lemma 4.10 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 [41]. For the second condition, we need to bound transition probabilities for the potential. Owing to our pessimistic assumption, the number of ones can only increase by at most 1.

For jumps decreasing the number of ones, we need to argue more carefully. Let $i = |x_t|_1 \geq a$ be the current number of ones and let $p_{i,j}$ be the probability that $|x_{t+1}|_1 = i - j$. Note that for a jump back to the first slope it is sufficient that one offspring has at most $2n/3$ ones. A necessary requirement is that j bits flip, which has probability at most $1/(j!)$. By a union bound over λ offspring, $p_{i,j} \leq \lambda/(j!) \leq 2^{-\kappa/2} \cdot (\kappa/2)!/(j!)$ using our bound on λ. For $j \geq \kappa/2$ (as $\kappa \geq 2$), we have $j! \geq (\kappa/2)! \cdot 2^{j-\kappa/2}$ and $p_{i,j} \leq 2^{-j}$. This implies for all $i \geq a$ and all j with $i - j \leq 2n/3$:

$$\Pr\left(|x_t|_1 - |x_{t+1}|_1 \geq j \right) \leq \sum_{j' \geq j} 2^{-j'} \leq 2 \cdot 2^{-j}.$$

In particular, $p_{i,\lambda}^{\uparrow} \leq \sum_{j=\kappa/2}^{n-i} p_{i,j} \leq \sum_{j=\kappa/2}^{\infty} 2 \cdot 2^{-j} = 4 \cdot 2^{-\kappa/2}$.

For $i - j > 2n/3$ the number of ones only decreases by j if *all* offspring decrease their number of ones by at least j, or if there is one offspring on the first slope. The probability of all offspring decreasing their number of ones by j is bounded by the probability that the first offspring decreases its number of ones by j. This is bounded by the probability of j bits flipping, which is at most $1/(j!) \leq 2/2^j$. Hence,

$$\forall (i-j) > 2n/3 : \ \Pr\left(|x_t|_1 - |x_{t+1}|_1 \geq j \right) \leq 2 \cdot 2^{-j} + p_{i,\lambda}^{\uparrow} \leq 6 \cdot 2^{-j}.$$

The possible penalty in the definition of g changes by at most $\max\left(\frac{2s}{s+1}, \frac{2s}{s+1} \cdot \frac{1}{s} \right) = \frac{2}{s+1} < 2$. Hence, for all t,

$$\Pr\left(|g(X_{t-1}) - g(X_t)| \geq j + 2 \mid g(X_t) > a \right) \leq \frac{24}{2^{j+2}},$$

which meets the second condition of Theorem 2 in [41] when choosing $\delta := 1$ and $r(\ell) := 24$.

The negative drift theorem [40,41] now implies that there exists a constant c^* such that the probability of the number of ones dropping below a in $2^{c^*(b-a)/24} \geq 2^{c^*\kappa/72}$ generations (or reaching a search point with at least $3n/4$ ones) is $2^{-\Omega((b-a)/24)} = 2^{-\Omega(\kappa)}$. Choosing $\gamma := \min\{2^{c^*/72}, 2\}$, this is at least γ^κ generations and a probability of $\gamma^{-\Omega(\kappa)}$ as claimed. Taking a union bound over this failure probability and that from Lemma 4.11 proves the claim. □

Now we show that with probability $\Omega(1)$ a search point with at least $3n/4$ ones is reached, without returning to the first slope and without resetting λ. Thus, with the claimed probability the algorithm behaves as the self-adjusting $(1, \lambda)$ EA from [26] on OneMax throughout this part of the run.

Lemma 4.13. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 4.1. Assume the conditions from Lemma 4.12 hold for constants γ and $\kappa := \frac{\log\log n}{\log\log\log n}$. Then with probability $\Omega(1)$ a search point with at least $3n/4$ ones is reached within $O(n)$ generations.*

Moreover, with the claimed probability the algorithm does not go back to the first slope and does not reset λ before a search point with at least $3n/4$ ones is reached.

Proof. The statement of Lemma 4.12 satisfies the preconditions of Lemma 4.10 for $d = \kappa$. Then Lemma 4.10 implies a positive drift

$$\mathrm{E}\left(g(X_{t+1}) - g(X_t) \mid X_t \right) \geq \frac{1-s}{4e} =: \delta$$

for the next γ^κ generations, unless a search point with at least $3n/4$ ones has been reached. In the latter case we are done, hence we assume that the drift is bounded from below as stated through the next $t_\kappa := \min\{\gamma^\kappa, n/\delta\}$ generations.

Now we aim to apply an additive drift theorem to bound the expected time to increase the potential by $D := \delta/12 \cdot \min\{\gamma^\kappa, n/\delta\} - 3$. To this end, we again assume that the number-of-ones component in the potential can only increase by at most 1. Since λ_t increases by a factor of $F^{1/s}$, the penalty term in Definition 4.9 can only increase by at most $2/(s+1) \leq 2$. Together, the potential increases by at most 3. To be able to apply Theorem 7 in [46], we consider the process $Y_t := g(X_0) + D - g(X_t)$ and the hitting time $T_Y := \inf\{t \mid Y_t \leq 0\}$, corresponding to decreasing the potential from its initial value $g(X_0)$ by at least D. As the Y-process decreases in expectation by δ, the preconditions of Theorem 7 in [46] are met. Since the potential can only increase by at most 3, the Y-process decreases by at most 3 and $\mathrm{E}\left(Y_T \mid X_0 \right) \geq -3$. Together, we obtain an upper bound of

$$\mathrm{E}\left(T_Y \mid Y_0 \right) \leq \frac{Y_0 - \mathrm{E}\left(Y_T \mid X_0 \right)}{\delta} \leq \frac{D+3}{\delta} = \frac{\min\{\gamma^\kappa, n/\delta\}}{12}.$$

By Markov's inequality, the probability that after t_κ steps the potential has not increased by D is at most $1/12$.

Assuming that the potential has increased by D, by Definition 4.9 the number of ones has increased by $D - O(1) = \min\{\delta/12 \cdot \gamma^{\kappa}, n/12\} - O(1)$. Since we start with at least $2n/3 + \kappa = 2n/3 + \omega(1)$ ones, there is a generation amongst the next γ^{κ} generations in which the number of ones is at least $\min\{2n/3 + \delta/12 \cdot \gamma^{\kappa}, 3n/4\}$.

As motivated earlier, we now iterate the above arguments a constant number of times, for exponentially increasing values of κ. Let $\kappa_0 := \kappa = \frac{\log\log n}{\log\log\log n}$ and define $\kappa_i := \delta/12 \cdot \gamma^{\kappa_{i-1}}$ for $i > 0$. Note that we have just showed that we have found a search point with at least $\min\{\kappa_1, 3n/4\}$ ones. If the number of ones is less than $3n/4$, the current state (x_t, λ_t) is κ_1-safe as it is weakly κ_0-safe and so $\lambda_t \le 2^{-\kappa_0/2} \cdot (\kappa_0/2)! \le 2^{-\kappa_1/2} \cdot (\kappa_1/2)!$.

Iterating the above argument with κ_1 instead of κ_0, we find a search point with at least $\min\{2n/3 + \kappa_2, 3n/4\}$ ones within the next $\min\{\gamma^{\kappa_1}, n/\delta\}$ generations, with probability at least $1 - 1/12 - \gamma^{-\Omega(\kappa_1)}$. We again iterate the argument with κ_2. We claim that $t_{\kappa_2} := \min\{\gamma^{\kappa_2}, n/\delta\} = n/\delta$ and show this by bounding κ_1, κ_2 and κ_3 from below.

$$\kappa_1 = \frac{\delta}{12}\gamma^{\frac{\log\log n}{\log\log\log n}} = 2^{\log(\gamma)\cdot \frac{\log\log n}{\log\log\log n} + \log(\delta/12)}$$
$$\ge 2^{\log\left(\frac{2}{\log(\gamma)}\log\log n\right)} = \frac{2}{\log(\gamma)}\log\log n.$$

Now, κ_2 is at least

$$\kappa_2 = \frac{\delta}{12}\gamma^{\kappa_1} \ge \frac{\delta}{12}\gamma^{\frac{2}{\log(\gamma)}\log\log n} = \frac{\delta}{12}\log^2 n \ge \frac{2}{\log(\gamma)}\log n$$

for n large enough. Likewise,

$$\kappa_3 = \frac{\delta}{12}\gamma^{\kappa_2} \ge \frac{\delta}{12}\cdot\gamma^{\frac{2}{\log(\gamma)}\log n} = \frac{\delta}{12}\cdot n^2.$$

Together, we have that within $\min\{\gamma^{\kappa_0}, n/\delta\} + \min\{\gamma^{\kappa_1}, n/\delta\} + \min\{\gamma^{\kappa_2}, n/\delta\} = O(n)$ generations, with probability at least $1 - \frac{3}{12} - \gamma^{-\Omega(\kappa_0)} - \gamma^{-\Omega(\kappa_1)} - \gamma^{-\Omega(\kappa_2)} \ge \frac{3}{4} - 3\gamma^{-\Omega(\kappa_0)} = \Omega(1)$ we have reached a search point with at least $3n/4$ ones, without going back to the first slope. The probability of a reset during $O(n)$ expected generations is exponentially small by Lemma 4.2 and (9), hence this failure probability can be absorbed in the $\Omega(1)$ probability bound. This completes the proof. $\square$

4.4. Finding the global optimum

Once the self-adjusting $(1, \lambda)$ EA moves far away from the cliff, the probability of jumping back up the cliff is reduced, and the next part of the optimisation resembles OneMax. The algorithm still can reset λ to 1. Such a steep decrease of λ would typically make the algorithm slip down the second slope until λ recovers to large enough values that support hill climbing. Hence, resets would break the runtime analysis made in [26]. We show in Lemma 4.14 that, with probability $\Omega(1)$, the algorithm neither jumps back up the cliff nor resets λ during the last part of the optimisation. This allows us to apply the previous analysis from [26] on OneMax.

Lemma 4.14. *Consider the self-adjusting $(1, \lambda)$ EA as in Theorem 4.1. For any initial $\lambda_0 \le \lambda_{\max}$ with $\lambda_0 = O(n \log n)$ and any initial search point x_0 with $|x_0|_1 \ge 3n/4$ the probability that the self-adjusting $(1, \lambda)$ EA creates the optimum without jumping back up the cliff or resetting λ to 1 is at least $1 - \frac{1}{e-1} - O\left(\frac{\log^3(n)}{n}\right)$.*

Proof. As long as the self-adjusting $(1, \lambda)$ EA does not jump back up the cliff, the self-adjusting $(1, \lambda)$ EA behaves as the self-adjusting $(1, \lambda)$ EA on OneMax. Additionally, if it does not have an unsuccessful generation with $\lambda = \lambda_{\max}$ it will never reset to 1, behaving as the self-adjusting $(1, \lambda)$ EA studied in [26].

From [26, Theorem 3.1 and Theorem 3.5] we know that the self-adjusting $(1, \lambda)$ EA solves OneMax in expected $O(n)$ generations and $O(n \log n)$ evaluations. Therefore, within these expected times our algorithm either finds the global optimum, jumps back up the cliff or resets λ to 1. We show that with probability $\Omega(1)$, a global optimum is reached.

In order for λ to reset at the same time as there is a jump back up the cliff, at least one offspring must flip $n/3$ one-bits and all other offspring must not increase their fitness. The probability of flipping $n/3$ bits is $n^{-\Omega(n)}$, hence the probability of both events happening at the same time is at most $n^{-\Omega(n)}$.

By Lemma 4.7 if the initial search point x_0 has $|x_0|_1 \ge 3n/4$, with probability $1 - O(1/n)$ the number of one-bits will never drop below $3n/4 - O(\log n)$ before finding the optimum or resetting λ. This means that, for the algorithm to jump back up the cliff at least one offspring must flip a linear amount of bits. The probability that one offspring flips a linear amount of bits is $n^{-\Omega(n)}$. By (9), the probability that this happens during $O(n \log n)$ expected evaluations is still $n^{-\Omega(n)}$. In the following we assume that we never return to the first slope.

To show that there is never an unsuccessful generation with $\lambda = \lambda_{\max}$ (i.e. λ never resets to 1) we divide the optimisation in two phases. The first phase ends the first time a state (x_t, λ_t) is found with $\lambda_t \ge 4 \log n$ and $|x_t|_1 \ge n - 3 \ln n$ or the optimum is found, and the second phase ends when the optimum is found.

During the first phase, we have $\lambda_t < 4\log n$ or $|x_t|_1 < n - 3\ln n$. The former condition implies $\lambda_t < \lambda_{\max}$ and resets are impossible, hence a reset in the first phase can only happen if $|x_t|_1 < n - 3\ln n$. In order to reach $\lambda = \lambda_{\max}$ at least one generation with $\lambda \geq en$ must be unsuccessful. The probability of an unsuccessful generation with $\lambda \geq en$ is at most

$$1 - p^{\rightarrow}_{i,\lambda} + p^{\uparrow}_{i,\lambda} \leq \left(1 - \frac{n-i}{en}\right)^{en} \leq \left(1 - \frac{3\ln n}{en}\right)^{en} \leq e^{-3\ln n} = n^{-3}.$$

Given that the optimum is found after $O(n)$ expected generations, by (9) the probability of reaching $\lambda = \lambda_{\max}$ during the first phase is $O(1/n^2)$.

For the second phase we first argue that the current fitness does not decrease, with high probability. The second phase starts with $\lambda \geq 4\log n$ and by Lemma 4.7 while $\lambda_t \geq 4\log n$ the fitness is not reduced before reaching the optimum with probability $1 - O(1/n)$. We now show that $\lambda \geq 4\log n$ throughout the remainder of the run with high probability.

Given the assumption that we never return to the first slope, by Lemma 3.6 in [26] from $|x|_1 \geq n - 3\ln n$ in expectation the optimum will be reached in $O(\log n)$ generations. To reduce λ to a value smaller than $4\log n$, a generation with $\lambda < 4F\log n$ must be successful. This event has a probability of at most

$$1 - \left(\frac{i}{n}\right)^{\lambda} \leq 1 - \left(1 - \frac{3\ln n}{n}\right)^{4F\log n} \leq \frac{12F\log^2 n}{n\log e} = O\left(\frac{\log^2(n)}{n}\right).$$

By (9) the probability that λ is reduced to a value less than $4\log n$ during the next $O(\log n)$ generations is $O\left(\frac{\log^3(n)}{n}\right)$. Accounting for both failures with probability $1 - O\left(\frac{\log^3(n)}{n}\right)$ each fitness value is left at most once.

Now we can calculate the probability of resetting λ by considering at most one generation with $\lambda = \lambda_{\max}$ per fitness value. We only have a reset of λ if one such generation is unsuccessful. Thus, the probability of resetting λ during the second phase is at most

$$\sum_{i=n-3\ln n}^{n-1} \left(1 - p^{+}_{i,\lambda_{\max}}\right) \leq \sum_{i=n-3\ln n}^{n-1} \left(1 - \frac{n-i}{en}\right)^{enF^{1/s}}$$
$$\leq \sum_{i=n-3\ln n}^{n-1} e^{-F^{1/s}(n-i)} = \sum_{j=1}^{3\ln n} e^{-F^{1/s}j} \leq \sum_{j=1}^{\infty} e^{-F^{1/s}j} = \frac{1}{1-e^{-F^{1/s}}} - 1 = \frac{1}{e^{F^{1/s}}-1} \leq \frac{1}{e-1}.$$

Adding up all failure probabilities completes the proof. □

4.5. Putting things together

Now we are able to prove the claimed bounds of $O(n)$ expected generations and $O(n\log n)$ expected evaluations from Theorem 4.1.

Proof of Theorem 4.1. From any initial state, by Lemma 4.3 we reach a solution x_t with $|x_t|_1 \geq 2n/3$ in expected $O(n)$ generations and $O(n + \lambda_{\max})$ evaluations.

Then, by Lemma 4.5, the algorithm reaches a κ-safe state (for $\kappa := \frac{\log\log n}{\log\log\log n}$) or a search point with at least $3n/4$ ones in $O(\log(\lambda_{\max})\log n)$ expected generations and $O(\lambda_{\max}\log n)$ expected evaluations.

Together, along with $\lambda_{\max} = \Omega(n)$ and $\lambda_{\max} = \text{poly}(n)$, the total time to reach a κ-safe state or a search point with at least $3n/4$ ones from an arbitrary initial state is $O(\log(\lambda_{\max})\log n + n) = O(n)$ expected generations and $O(\lambda_{\max}\log n)$ expected evaluations.

Now, we define a *trial* as a sequence of generations that starts in a κ-safe state and ends as soon as one of the following events happens: a search point with $|x_t|_1 \geq 3n/4$ is found, λ is reset or the algorithm jumps back to the first slope. Let T_{trial} denote the corresponding number of generations. By Lemma 4.13 $T_{\text{trial}} = O(n)$ with probability $\Omega(1)$. By definition, during all T_{trial} generations, the algorithm behaves like the self-adjusting $(1,\lambda)$ EA without resetting on OneMax (with the possible exception of the final selection step, which can lead the algorithm back to the first slope) and we can obtain an upper bound for the number of evaluations spent during the trial of $O(n\log n)$ from [26]. (Note that the additional events ending a trial can only decrease the number of evaluations, compared to a run of the original self-adjusting $(1,\lambda)$ EA on OneMax.) Then, by Lemma 4.13 at the end of the trial the algorithm reaches a solution with $|x_t|_1 \geq 3n/4$ with probability $\Omega(1)$. If instead the trial finishes by resetting λ or jumping back to the first slope we start the above arguments from scratch. By the arguments above the algorithm will reach a κ-safe point in $O(n)$ expected generations and $O(\lambda_{\max}\log n)$ expected evaluations and from there we start another trial. In expectation, a constant number of trials suffices to find a search point with at least $3n/4$ ones. Hence, from any initial state, in expectation in $O(n)$ generations and $O(\lambda_{\max}\log n)$ evaluations we reach a search point with at least $3n/4$ ones.

Likewise, from a search point with at least $3n/4$ ones, by Lemma 4.14 with probability $\Omega(1)$ we find the optimum without resetting λ or returning to the first slope, and hence the analysis from [26] still applies when considering a notion of a trial adapted to the aforementioned events. Thus, in expected $O(n)$ generations and $O(n\log n)$ evaluations we either reach the global optimum, return to the first slope or reset λ, and the probability of reaching the optimum is $\Omega(1)$. Iterating this argument an expected constant number of times proves the claimed bound. □

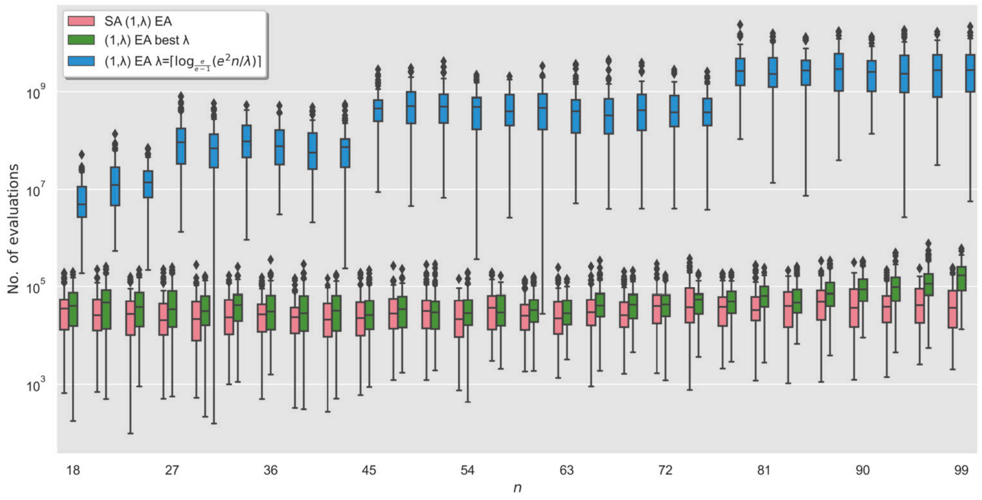

Fig. 2. Box plots of the number of evaluations used by the $(1, \lambda)$ EA with $\lambda = \left\lceil \log_{\frac{e}{e-1}}(e^2 n/\lambda) \right\rceil$ and the best λ (manually tuned), and the self-adjusting $(1, \lambda)$ EA with $s = 1$, $F = 1.5$, $\lambda_{\max} = enF^{1/s}$ on Cliff with $n \in \{18, 21, \dots, 99\}$ over 100 runs. The y-axis (number of evaluations) is log-scaled.

5. Experiments

In this section we conduct an experimental analysis to improve our understanding of the $(1, \lambda)$ EA with static λ and the self-adjusting $(1, \lambda)$ EA resetting λ on Cliff and other functions. All the experiments were performed using the IOHProfiler [47].

For algorithm comparisons we performed Mann-Whitney U Test for all pairs of algorithms compared in the text. We performed two-sided tests to check whether the two input distributions differ or not, followed by one-sided tests both ways to confirm which algorithm is stochastically faster than the other. We report a comparison as statistically significant if the p-values of the two-sided test and that of the respective one-sided test both satisfied $p \le 0.01$, that is, a confidence level of 0.01.

5.1. *Cliff*

We begin our analysis with a runtime comparison between the self-adjusting $(1, \lambda)$ EA resetting λ and the $(1, \lambda)$ EA with static λ on Cliff. For the $(1, \lambda)$ EA with static λ we use two values of λ. The first one is $\lambda = \left\lceil \log_{\frac{e}{e-1}}(e^2 n/\lambda) \right\rceil$ which is the smallest λ-value that theoretically guarantees an efficient runtime on OneMax [35].[3] The second value of λ used was manually tuned to the λ-value that yielded the best performance in terms of number of evaluations for each problem size n. For the self-adjusting $(1, \lambda)$ EA we used as parameters $s = 1$, $F = 1.5$ and $\lambda_{\max} = enF^{1/s}$. Fig. 2 displays box plots of the number of evaluations used by the three algorithms over 100 runs for different problem sizes on Cliff.

There are two main things we can appreciate from Fig. 2. Our first observation is that for the $(1, \lambda)$ EA with static λ, there are smaller λ-values than the theoretical best parameter values which perform better for the problem sizes tested here. This is because the exponential lower bound (from [2, Theorem 10]) for the runtime of the $(1, \lambda)$ EA when using $\lambda \le (1 - \varepsilon) \log_{\frac{e}{e-1}} n$ only becomes important for large values of n. This observation was also made by Rowe and Sudholt [2] for OneMax with $n = 10000$. Our second observation is that even with the best (tuned) static λ-value the self-adjusting $(1, \lambda)$ EA shows a slightly better performance, where the performance gap seems to widen with n. However, the self-adjusting $(1, \lambda)$ EA is statistically faster only for $n \ge 87$.

Because of this last observation we ran more experiments for larger problem sizes comparing only the $(1, \lambda)$ EA with tuned λ and the self-adjusting $(1, \lambda)$ EA. The results of these experiments, shown in Fig. 3, demonstrate that for larger values of n the $(1, \lambda)$ EA with static λ is statistically slower than the self-adjusting $(1, \lambda)$ EA for all $n \ge 120$. In addition we observe an increasing trend for the number of evaluations needed by the $(1, \lambda)$ EA. Since the number of evaluations is normalised by $n \log n$, this indicates that the runtime of the $(1, \lambda)$ EA grows faster than $n \log n$. This means that the larger the problem size n, the worse the performance of the $(1, \lambda)$ EA with static λ is, compared to the self-adjusting $(1, \lambda)$ EA.

In Fig. 4 we explore how the $(1, \lambda)$ EA and self-adjusting $(1, \lambda)$ EA behave on the generalised Cliff function (Cliff$_d$) defined as:

$$\text{Cliff}_d(x) := \begin{cases} |x|_1 & \text{if } |x|_1 \le n - d, \\ |x|_1 - d + 1/2 & \text{otherwise.} \end{cases}$$

[3] Strictly speaking, we would need to use $\lambda := \left\lceil \log_{\frac{e}{e-1}}(cn/\lambda) \right\rceil$ for an arbitrary constant $c > e^2$. Since $\log_{\frac{e}{e-1}}(e^2 n/\lambda)$ is not an integer, there is a sufficiently small constant c with $c > e^2$ such that replacing e^2 with c does not change the integer returned by the ceiling function for all values of λ and n investigated empirically here.

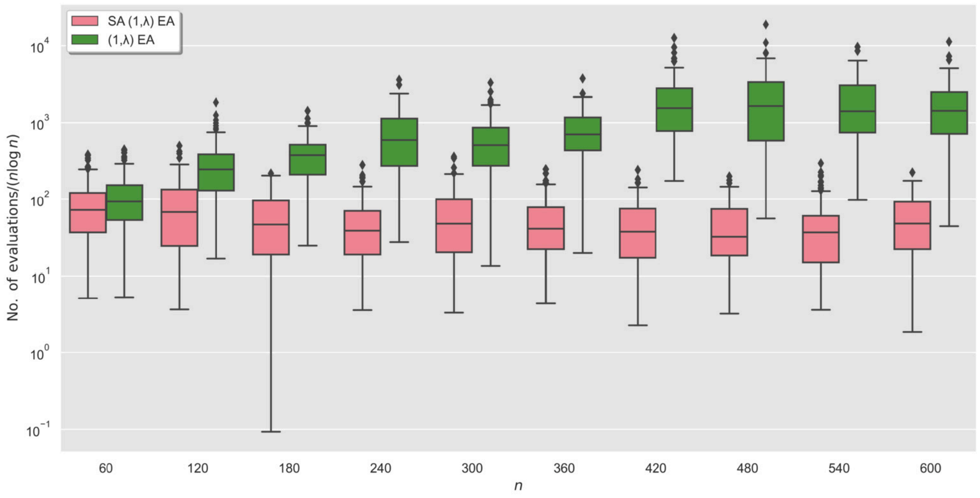

Fig. 3. Box plots of the number of evaluations used by the $(1,\lambda)$ EA with the best λ (manually tuned), and the self-adjusting $(1,\lambda)$ EA with $s=1$, $F=1.5$, $\lambda_{\max}=enF^{1/s}$ on Cliff over 100 runs. The y-axis (number of evaluations) is normalised by $n\log n$ and log-scaled.

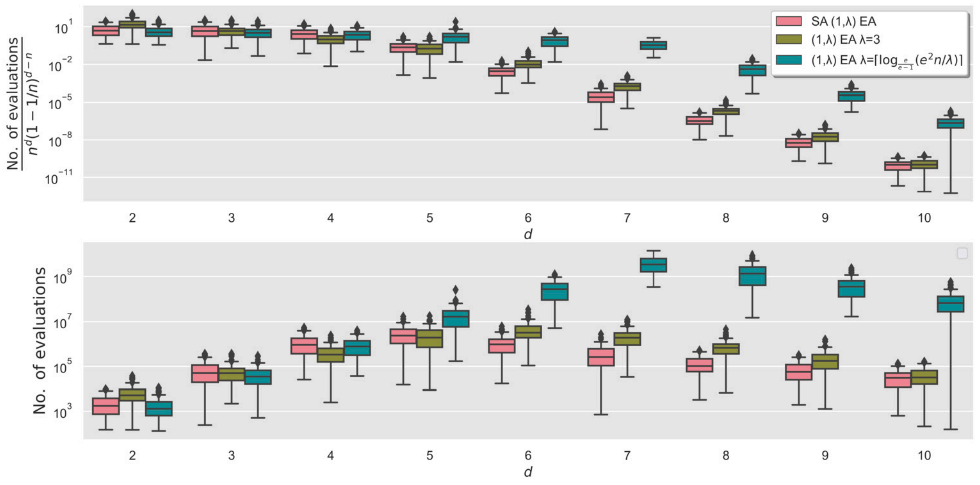

Fig. 4. Box plots of the number of evaluations used by the $(1,\lambda)$ EA with $\lambda=\left\lceil\log_{\frac{e}{e-1}}(e^2n/\lambda)\right\rceil$ and $\lambda=3$ (best λ for the standard Cliff function), and the self-adjusting $(1,\lambda)$ EA with $s=1$, $F=1.5$, $\lambda_{\max}=enF^{1/s}$ on Cliff with $n=30$, varying the distance d of the cliff from the optimum over 100 runs. The y-axis (number of evaluations) is log-scaled and in the top box plots also normalised by $n^d(1-1/n)^{d-n}$.

In these experiments we use $n=30$ and different distance d from the *cliff* to the optimum. For the $(1,\lambda)$ EA we use $\lambda=\left\lceil\log_{\frac{e}{e-1}}(e^2n/\lambda)\right\rceil$ and $\lambda=3$ which is the best λ found by manual tuning for the standard Cliff function, that is, Cliff_d with $d=n/3$. Both plots in Fig. 4 show the same experiments, but the number of evaluations in the top box plots of Fig. 4 is normalised by $n^d(1-1/n)^{d-n}$, which is the asymptotic runtime of the $(1+1)$ EA on Cliff_d with distance d from the cliff to the optimum [31].

From Fig. 4 we can see that when the distance d from the cliff to the optimum is reduced, the problem becomes harder for all algorithms up to a maximum from which reducing d more reduces their runtime. We also see that for small d the algorithms' runtime seem to grow at almost $\Theta(n^d(1-1/n)^{n-d})$. Because of this, we conjecture that for small distances d from the cliff to the optimum most of the time the algorithms find the global optimum without jumping down the cliff, therefore the algorithms do not benefit from their non-elitist selection in these cases. On the other hand for larger values of d the algorithms benefit greatly from their non-elitist selection because they are able to jump down the cliff more easily. This leaves a d-value in the middle for which both jumping directly to the optimum and jumping down the cliff is difficult.

5.2. Varying $\lambda_{\max}$

We now explore how the hyper-parameter $\lambda_{\max}$ affects the performance of the self-adjusting $(1,\lambda)$ EA. Our theoretical analysis suggests that increasing the value of $\lambda_{\max}$ will increase the runtime of the algorithm because the algorithm spends more evaluations on the local optimum before resetting λ and therefore on jumping down the cliff. On the other hand decreasing the value of $\lambda_{\max}$ can

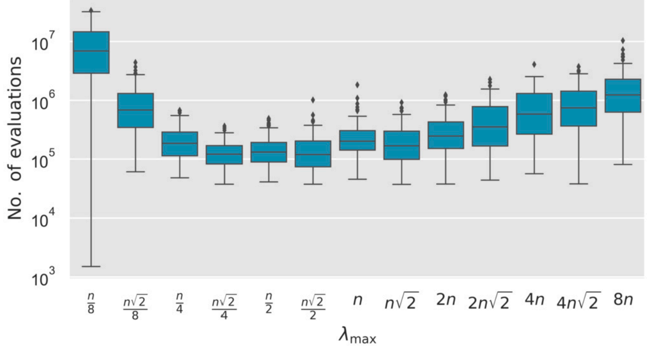

Fig. 5. Box plots of the number of evaluations used by the self-adjusting $(1, \lambda)$ EA with $s = 1$, $F = 1.5$ over 100 runs for different $\lambda_{\max}$ on CLIFF with $n = 1500$. The y-axis (number of evaluations) is log-scaled.

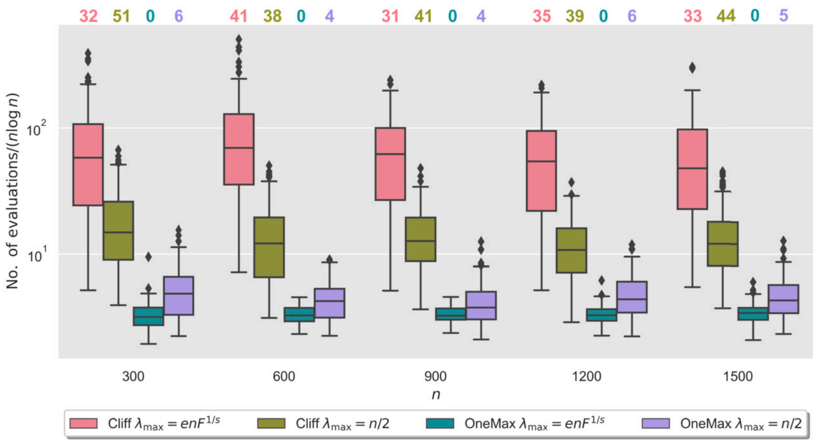

Fig. 6. Box plots of the number of evaluations used by the self-adjusting $(1, \lambda)$ EA with $s = 1$, $F = 1.5$ over 100 runs for $\lambda_{\max} \in \{enF^{1/s}, n/2\}$ on CLIFF and ONEMAX. The rounded average number of resets per run is shown on the top. The y-axis (number of evaluations) is normalised by $n \log n$ and log-scaled.

decrease the runtime of the algorithm until a point where resets in λ does not allow the algorithm to optimise the second slope. In our theoretical analysis we used a value of $\lambda_{\max} = enF^{1/s}$ that allows the algorithm to reset relatively fast when encountering a local optimum but also allows the algorithm to optimise the second slope with a small number of resets. We acknowledge that the factor $eF^{1/s}$ in our theoretical analysis was used to ease calculations; a smaller factor could improve performance, but only by a constant factor.

In Fig. 5 we show box plots of the number of evaluations used by the self-adjusting $(1, \lambda)$ EA with $s = 1$, $F = 1.5$ over 100 runs for different $\lambda_{\max}$ on CLIFF with $n = 1500$. As expected, larger values of $\lambda_{\max}$ increase the runtime of the algorithm. We can also see that there are smaller values than $\lambda_{\max} = enF^{1/s} \approx 4.08n$ that are better on CLIFF. Values between $\frac{n\sqrt{2}}{4}$ and $\frac{n\sqrt{2}}{2}$ obtain the best performance and reducing $\lambda_{\max}$ further than this deteriorates the performance of the algorithm substantially.

In Fig. 6 we show box plots of the number of evaluations used by the self-adjusting $(1, \lambda)$ EA with $s = 1$, $F = 1.5$ over 100 runs for $\lambda_{\max} \in \{enF^{1/s}, n/2\}$ on CLIFF and ONEMAX. On top of the plot we give the average number of resets per run rounded to the nearest integer. We can see that for all problem sizes n tested using $\lambda_{\max} = n/2$ gives an advantage to the self-adjusting $(1, \lambda)$ EA on CLIFF but a disadvantage on ONEMAX. This is because for $\lambda_{\max} = n/2$ the algorithm jumps down the cliff faster but there tends to be an increased number of resets on average. Since this happens in both ONEMAX and CLIFF, we attribute this to resets near the optimum where λ grows fast. An interesting observation is that using $\lambda = enF^{1/s}$ for ONEMAX allows the algorithm to find the optimum with almost no resets. Out of the 500 runs for all n there were only 5 resets and 3 of them were with $n = 300$. This suggests that our theoretical analysis was pessimistic, since we bounded the number of resets near the optimum in a typical run by a constant.

In Fig. 7 we show a detailed view of four example runs. Here we can see that on CLIFF both values of $\lambda_{\max}$ reset λ and jump down the cliff several times before starting to optimise the second slope. But, when using $\lambda_{\max} = n/2$ the fitness stagnates during the final generations before reaching the global optimum in both ONEMAX and CLIFF. This is because λ resets and then there are fallbacks in fitness. It is important to note that these fallbacks are small and the algorithm is able to recover fast from them. In our theoretical analysis we assumed a much worse case where a reset near the optimum would cause the algorithm to go back to the first slope.

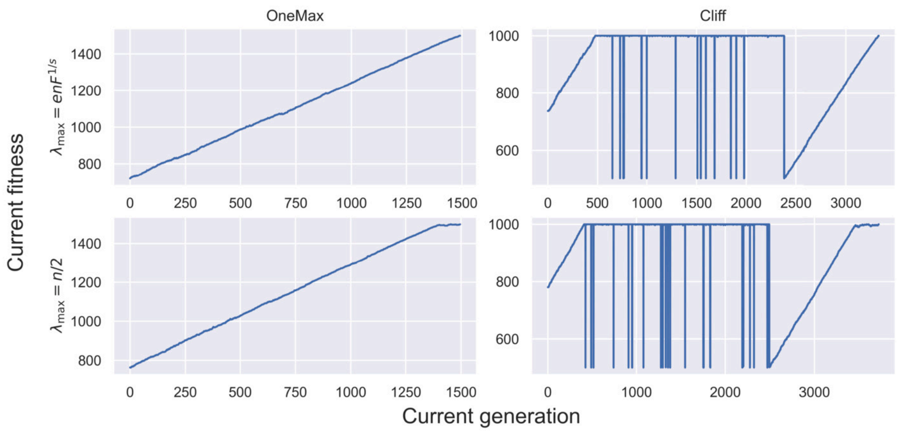

Fig. 7. Current fitness per generation of four example runs of the self-adjusting $(1, \lambda)$ EA with $\lambda_{\max} = enF^{1/s}$ (top) and $\lambda_{\max} = n/2$ (down) on OneMax (left) and Cliff (right) with $n = 1500$.

5.3. Other problems

We finish our empirical analysis with a comparison between the self-adjusting $(1, \lambda)$ EA and the $(1, \lambda)$ EA with static λ for different problems. We select four problems from the Pseudo-Boolean Optimization (PBO) problem set proposed by [48].

- Low Autocorrelation Binary Sequences (LABS): A non-linear combinatorial problem. The problem is to maximise the merit factor of a binary sequence. The merit factor is proportional to the reciprocal of the sequence's autocorrelation (also called serial correlation). Its equivalent pseudo-Boolean function is:

$$\mathrm{LABS}(x) = \frac{n^2}{2\sum_{k=1}^{n-1}\left(\sum_{i=1}^{n-k} x'_i \cdot x'_{i+k}\right)^2} \quad \text{where } x'_i = 2x_i - 1.$$

- Ising Model problem: consists of discrete variables that represent magnetic dipole moments of atomic "spins" that can be in one of two states (+1 or −1). The spins are arranged in a graph describing the interactions strengths (edges) between spins (vertices), here we use a two-dimensional torus lattice. Neighbouring spins with the same state have a lower interaction than those with a different state. The ISG problem is to set the signs of all spins to minimise interactions.
- Maximum Independent Vertex Set (MIS): An independent vertex set is a subset of vertices where no vertex in the subset is adjacent to any other vertex within the subset. The MIS problem is to find the largest independent vertex set for a given graph.
- N-Queens problem: Is the task to place N queens in a chess board of size $N \times N$ in such a way that the N queens cannot "attack" each other.

For more information on the problems we refer the reader to [48] and the IOHProfiler project [47]. We performed 100 runs of the self-adjusting $(1, \lambda)$ EA with $\lambda_{\max} \in \{enF^{1/s}, n\}$ and the $(1, \lambda)$ EA with $\lambda = \lceil \log_{\frac{e}{e-1}} n \rceil$ across the four problems with different problem sizes. Fig. 8 shows the results of these runs. We included the self-adjusting $(1, \lambda)$ EA with $\lambda_{\max} = n$ because the results from Section 5.2 suggest that the factor $eF^{1/s}$ used in our theoretical analysis is not needed, but a smaller value of $\lambda_{\max} = n/2$ already results in a substantial number of resets.

Overall we can see that the self-adjusting $(1, \lambda)$ EA with $\lambda_{\max} = n$ performs slightly better than its counterpart with $\lambda_{\max} = enF^{1/s}$ for all problems and most problem sizes tested. When comparing the self-adjusting $(1, \lambda)$ EA with $\lambda_{\max} = n$ and the $(1, \lambda)$ EA with static λ we found that the self-adjusting $(1, \lambda)$ EA is statistically significantly faster than the $(1, \lambda)$ EA for most problem sizes on LABS, MIS and the N-Queens problem, but statistically significantly slower on Ising models. The main conclusion that we draw from Fig. 8 is that the self-adjusting $(1, \lambda)$ EA is able to perform better or similar to the $(1, \lambda)$ EA on common multimodal benchmark problems with the added advantage of easier parameter selection.

6. Conclusions

The usefulness of parameter control has so far mainly been demonstrated for elitist EAs on relatively easy problems. For the more difficult multimodal problem Cliff we showed that the self-adjusting $(1, \lambda)$ EA using success-based rules and a reset mechanism can find the global optimum in $O(n)$ expected generations and $O(n \log n)$ expected evaluations. This is a speedup of order $\Omega(n^{2.9767})$ over the expected optimisation time with the best fixed value of λ, up to sub-polynomial terms.

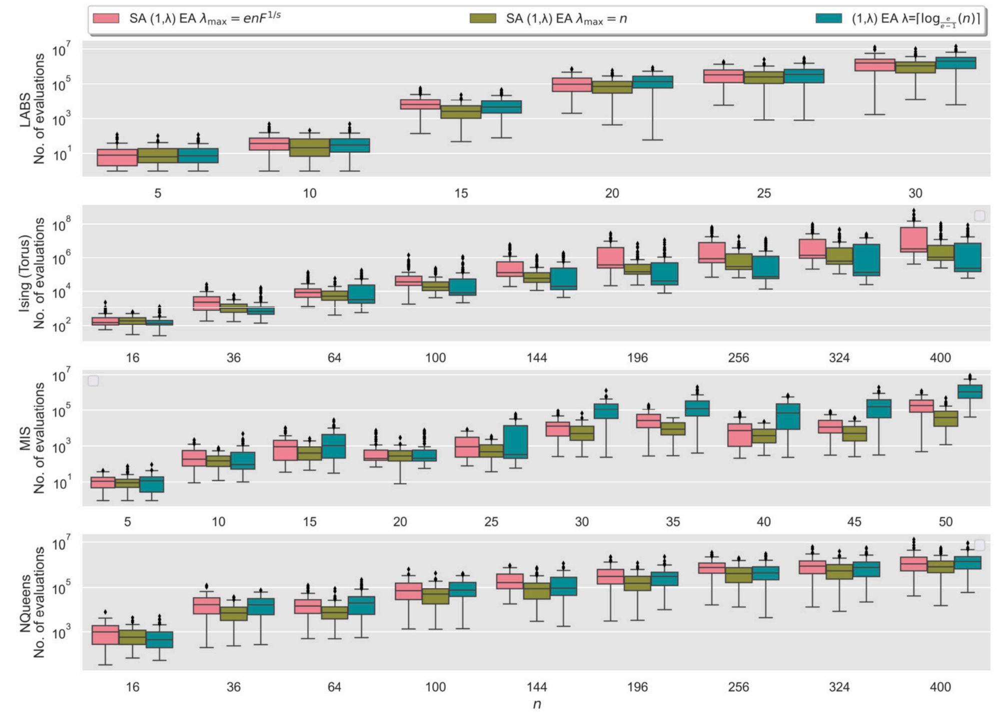

Fig. 8. Box plots of the number of evaluations used by the self-adjusting $(1, \lambda)$ EA with $s = 1$, $F = 1.5$ over 100 runs for $\lambda_{\max} \in \{enF^{1/s}, n\}$ on LABS, Ising, MIS and N-Queens problems. The y-axis (number of evaluations) is log-scaled.

The latter conclusion was obtained by refining the previous bounds on the expected optimisation time of the $(1, \lambda)$ EA on CLIFF from [30], $O(e^{5\lambda}) = O(148.413^{\lambda})$ and $\min\{n^{n/4}, e^{\lambda/4}\}/3 = \min\{n^{n/4}, 1.284^{\lambda}\}/3$, towards bounds of $\Omega(\xi^{\lambda})$ and $O(\xi^{\lambda} \log n)$, for $\xi \approx 6.196878$, revealing the degree of the polynomial in the expected runtime of the $(1, \lambda)$ EA with the best fixed λ as $\eta \approx 3.976770136$.

Our theoretical results demonstrate the power of parameter control for the multimodal CLIFF problem and that drastic performance improvement can be obtained. We extended our theoretical results with an extensive empirical analysis. We showed how the $(1, \lambda)$ EA and the self-adjusting $(1, \lambda)$ EA behave on CLIFF with realistic problem sizes and CLIFF functions where the position of the cliff is chosen differently from $2n/3$ ones. We also analysed how the hyper-parameter $\lambda_{\max}$ affects the performance of the algorithm, showing that too small values of $\lambda_{\max}$ can deteriorate the performance of the algorithm but $\lambda_{\max} = n$ and $\lambda_{\max} = enF^{1/s}$ achieve a good performance on all problems tested. Finally we considered other problems and showed that the self-adjusting $(1, \lambda)$ EA performs better or similarly to the $(1, \lambda)$ EA with the added advantage of easier parameter selection.

For the $(1, \lambda)$ EA the parameter λ has a huge impact on performance. If λ is too small we obtain exponential runtimes in the problem size n [2] and λ needs to be $\Omega(\log n)$ to have efficient runtimes. But the probability of accepting a worsening in fitness and leaving the local optima is related exponentially to the parameter λ, which means that increases in λ by a constant factor can increase the runtime by a polynomial factor in the problem size n, making parameter tuning a hard task. On the other hand for the self-adjusting $(1, \lambda)$ EA the parameter $\lambda_{\max}$ is less critical. Although small $\lambda_{\max}$ can deteriorate the performance of the algorithm, our theoretical and empirical analyses suggest that the time to escape the local optima depends only linearly on $\lambda_{\max}$ (instead of exponentially), therefore a wide range of $\lambda_{\max}$-values obtain the same asymptotic runtime. This eases the task of parameter tuning for the self-adjusting $(1, \lambda)$ EA.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Mario Alejandro Hevia Fajardo reports financial support was provided by National Council on Science and Technology.

Data availability

Data will be made available on request.

Acknowledgements

This research has been supported by CONACYT (Consejo Nacional de Ciencia y Tecnología) under the grant no. 739621.

References

[1] B. Doerr, C. Doerr, Theory of parameter control for discrete black-box optimization: provable performance gains through dynamic parameter choices, in: B. Doerr, F. Neumann (Eds.), Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 271–321.
[2] J.E. Rowe, D. Sudholt, The choice of the offspring population size in the $(1,\lambda)$ evolutionary algorithm, Theor. Comput. Sci. 545 (2014) 20–38.
[3] J. Lengler, A general dichotomy of evolutionary algorithms on monotone functions, IEEE Trans. Evol. Comput. 24 (2020) 995–1009.
[4] T. Friedrich, F. Neumann, What's hot in evolutionary computation, Proc. AAAI Conf. Artif. Intell. 31 (2017) 5064–5066.
[5] B. Doerr, F. Neumann, A survey on recent progress in the theory of evolutionary algorithms for discrete optimization, ACM Trans. Evol. Learn. Optim. 1 (2021).
[6] S. Böttcher, B. Doerr, F. Neumann, Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem, Proc. of PPSN XI, vol. 6238, Springer, 2010, pp. 1–10.
[7] G. Badkobeh, P.K. Lehre, D. Sudholt, Unbiased black-box complexity of parallel search, in: Proc. of PPSN XIII, Springer, 2014, pp. 892–901.
[8] J. Lässig, D. Sudholt, Adaptive population models for offspring populations and parallel evolutionary algorithms, in: Proc. of FOGA, ACM, 2011, pp. 181–192.
[9] B. Doerr, C. Doerr, F. Ebel, From black-box complexity to designing new genetic algorithms, Theor. Comput. Sci. 567 (2015) 87–104.
[10] B. Doerr, C. Doerr, Optimal static and self-adjusting parameter choices for the $(1+(\lambda,\lambda))$ genetic algorithm, Algorithmica 80 (2018) 1658–1709.
[11] M.A. Hevia Fajardo, D. Sudholt, Theoretical and empirical analysis of parameter control mechanisms in the $(1+\ \lambda,\lambda))$ genetic algorithm, ACM Trans. Evol. Learn. Optim. 2 (2023), https://doi.org/10.1145/3564755.
[12] B. Doerr, C. Gießen, C. Witt, J. Yang, The $(1+\ \lambda)$ evolutionary algorithm with self-adjusting mutation rate, Algorithmica 81 (2019) 593–631.
[13] A. Mambrini, D. Sudholt, Design and analysis of schemes for adapting migration intervals in parallel evolutionary algorithms, Evol. Comput. 23 (2015) 559–582.
[14] B. Doerr, C. Doerr, J. Lengler, Self-adjusting mutation rates with provably optimal success rules, Algorithmica 83 (2021) 3108–3147, https://doi.org/10.1007/s00453-021-00854-3.
[15] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, Simple hyper-heuristics control the neighbourhood size of randomised local search optimally for LeadingOnes, Evol. Comput. 28 (2020) 437–461.
[16] B. Doerr, A. Lissovoi, P.S. Oliveto, J.A. Warwicker, On the runtime analysis of selection hyper-heuristics with adaptive learning periods, in: Proc. of GECCO, ACM, 2018, pp. 1015–1022.
[17] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, How the duration of the learning period affects the performance of random gradient selection hyper-heuristics, Proc. AAAI 34 (2020) 2376–2383.
[18] A. Rajabi, C. Witt, Evolutionary algorithms with self-adjusting asymmetric mutation, in: Proc. of PPSN XVI, Springer, 2020, pp. 664–677.
[19] T. Jansen, D. Sudholt, Analysis of an asymmetric mutation operator, Evol. Comput. 18 (2010) 1–26.
[20] A. Rajabi, C. Witt, Self-adjusting evolutionary algorithms for multimodal optimization, Algorithmica 84 (2022) 1694–1723, https://doi.org/10.1007/s00453-022-00933-z.
[21] A. Rajabi, C. Witt, Stagnation detection with randomized local search, Evol. Comput. 31 (2023) 1–29, https://doi.org/10.1162/evco_a_00313.
[22] D.-C. Dang, P.K. Lehre, Self-adaptation of mutation rates in non-elitist populations, in: Proc. of PPSN XIV, Springer, Cham, 2016, pp. 803–813.
[23] B. Case, P.K. Lehre, Self-adaptation in nonelitist evolutionary algorithms on discrete problems with unknown structure, IEEE Trans. Evol. Comput. 24 (2020) 650–663.
[24] B. Doerr, C. Witt, J. Yang, Runtime analysis for self-adaptive mutation rates, Algorithmica 83 (2021) 1012–1053.
[25] A. Lissovoi, P.S. Oliveto, J.A. Warwicker, On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation, Proc. AAAI 33 (2019) 2322–2329.
[26] M.A. Hevia Fajardo, D. Sudholt, Self-adjusting population sizes for non-elitist evolutionary algorithms: why success rates matter, in: Proc. of GECCO, ACM, 2021, pp. 1151–1159.
[27] M.A. Hevia Fajardo, D. Sudholt, Self-adjusting population sizes for non-elitist evolutionary algorithms: why success rates matter, Algorithmica (2023), https://doi.org/10.1007/s00453-023-01153-9.
[28] M. Kaufmann, M. Larcher, J. Lengler, X. Zou, Self-adjusting population sizes for the $(1,\lambda)$-EA on monotone functions, ArXiv e-prints arXiv:2204.00531, 2022.
[29] D.-C. Dang, A. Eremeev, P.K. Lehre, Escaping local optima with non-elitist evolutionary algorithms, Proc. AAAI Conf. Artif. Intell. 35 (2021) 12275–12283.
[30] J. Jägersküpper, T. Storch, When the plus strategy outperforms the comma strategy and when not, in: Proc. of IEEE FOCI, IEEE, 2007, pp. 25–32.
[31] T. Paixão, J. Pérez Heredia, D. Sudholt, B. Trubenová, Towards a runtime comparison of natural and artificial evolution, Algorithmica 78 (2017) 681–713.
[32] P.K. Lehre, C. Witt, Black-box search by unbiased variation, Algorithmica 64 (2012) 623–642.
[33] D. Corus, P.S. Oliveto, D. Yazdani, When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms, Theor. Comput. Sci. 832 (2020) 166–185.
[34] M.A. Hevia Fajardo, D. Sudholt, Self-adjusting offspring population sizes outperform fixed parameters on the cliff function, in: Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA '21, ACM, 2021, pp. 5:1–5:15.
[35] J. Bossek, D. Sudholt, Do additional target points speed up evolutionary algorithms?, Theor. Comput. Sci. 950 (2023) 113757, https://doi.org/10.1016/j.tcs.2023.113757.
[36] B. Doerr, Probabilistic tools for the analysis of randomized optimization heuristics, in: B. Doerr, F. Neumann (Eds.), Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, Springer, 2020, pp. 1–87.
[37] J. He, X. Yao, A study of drift analysis for estimating computation time of evolutionary algorithms, Nat. Comput. 3 (2004) 21–35.
[38] C. Witt, Tight bounds on the optimization time of a randomized search heuristic on linear functions, Comb. Probab. Comput. 22 (2013) 294–318.
[39] G. Badkobeh, P.K. Lehre, D. Sudholt, Black-box complexity of parallel search with distributed populations, in: Proc. of FOGA, ACM, 2015, pp. 3–15.
[40] P.S. Oliveto, C. Witt, Simplified drift analysis for proving lower bounds in evolutionary computation, Algorithmica 59 (2011) 369–386.
[41] P.S. Oliveto, C. Witt, Erratum: simplified drift analysis for proving lower bounds in evolutionary computation, ArXiv e-prints arXiv:1211.7184, 2012.
[42] B. Doerr, C. Doerr, T. Kötzing, Static and self-adjusting mutation strengths for multi-valued decision variables, Algorithmica 80 (2018) 1732–1768.
[43] Y. Akimoto, A. Auger, T. Glasmachers, Drift theory in continuous search spaces: expected hitting time of the $(1+1)$-ES with 1/5 success rule, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '18, ACM, 2018, pp. 801–808.
[44] D. Morinaga, Y. Akimoto, Generalized drift analysis in continuous domain: linear convergence of (1+1)-ES on strongly convex functions with Lipschitz continuous gradients, in: Proceedings of the 15th ACM/SIGEVO Conference on Foundations of Genetic Algorithms, FOGA '19, ACM, 2019, pp. 13–24.
[45] D. Morinaga, K. Fukuchi, J. Sakuma, Y. Akimoto, Convergence rate of the (1+1)-evolution strategy with success-based step-size adaptation on convex quadratic functions, in: Proceedings of the Genetic and Evolutionary Computation Conference, GECCO '21, ACM, 2021, pp. 1169–1177.
[46] T. Kötzing, M.S. Krejca, First-hitting times under drift, Theor. Comput. Sci. 796 (2019) 51–69, https://doi.org/10.1016/j.tcs.2019.08.021, https://www.sciencedirect.com/science/article/pii/S0304397519305109.
[47] C. Doerr, H. Wang, F. Ye, S. van Rijn, T. Bäck, IOHprofiler: a benchmarking and profiling tool for iterative optimization heuristics, arXiv:1810.05281, 2018, https://arxiv.org/abs/1810.05281.
[48] C. Doerr, F. Ye, N. Horesh, H. Wang, O.M. Shir, T. Bäck, Benchmarking discrete optimization heuristics with IOHprofiler, Appl. Soft Comput. 88 (2020) 106027.