Automatic adaptation of hypermutation rates for multimodal optimisation

Corus, Dogan; Oliveto, Pietro S.; Yazdani, Donya

Published in:
FOGA 2021 - Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms

DOI:
10.1145/3450218.3477305

Publication date:
2021

Citation for published version (APA):
Corus, D., Oliveto, P. S., & Yazdani, D. (2021). Automatic adaptation of hypermutation rates for multimodal optimisation. In *FOGA 2021 - Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms* (FOGA 2021 - Proceedings of the 16th ACM/SIGEVO Conference on Foundations of Genetic Algorithms). Association for Computing Machinery, Inc. https://doi.org/10.1145/3450218.3477305

Automatic Adaptation of Hypermutation Rates for Multimodal Optimisation

Dogan Corus
Kadir Has University
Fatih, Istanbul, Turkey
dogan.corus@khas.edu.tr

Pietro S. Oliveto
The University of Sheffield
Sheffield, UK
p.oliveto@sheffield.ac.uk

Donya Yazdani
Aberystwyth University
Aberystwyth, UK
d.yazdani@aber.ac.uk

ABSTRACT

Previous work has shown that in Artificial Immune Systems (AIS) the best static mutation rates to escape local optima with the ageing operator are far from the optimal ones to do so via large hypermutations and vice-versa. In this paper we propose an AIS that automatically adapts the mutation rate during the run to make good use of both operators. We perform rigorous time complexity analyses for standard multimodal benchmark functions with significant characteristics and prove that our proposed algorithm can learn to adapt the mutation rate appropriately such that both ageing and hypermutation are effective when they are most useful for escaping local optima. In particular, the algorithm provably adapts the mutation rate such that it is efficient for the problems where either operator has been proven to be effective in the literature.

CCS CONCEPTS

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

KEYWORDS

randomized search heuristics, artificial immune systems, evolutionary algorithms, hypermutations, ageing, multimodal optimization, parameter adaptation, runtime analysis

ACM Reference Format:
Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2021. Automatic Adaptation of Hypermutation Rates for Multimodal Optimisation. In *Foundations of Genetic Algorithms XVI (FOGA '21), September 6–8, 2021, Virtual Event, Austria.* ACM, New York, NY, USA, 12 pages. https://doi.org/10.1145/3450218.3477305

1 INTRODUCTION

It is well understood that the performance of virtually all general purpose optimisation algorithms depends crucially on the parameter settings of their variation operators, used to create new candidate solutions, and of their selection operators used to decide whether to accept new solutions or keep previously identified ones. For instance the performance of local search algorithms [31] on a given problem class depends on the neighbourhood size choice, while for simulated annealing [27] also the cooling schedule used in the selection phase comes into play. Nature inspired optimisation algorithms often come with parameter ranges that are suggested by the underlying paradigm that they imitate. For example, traditional generational evolutionary algorithms (EAs) and genetic algorithms (GAs), inspired by Darwinian evolution, naturally require low mutation rates or their runtime will be exponential even for any function with unique optimum [4].

In recent years increasing evidence has been provided that steady-state EAs and GAs become more efficient either at hillclimbing or at escaping from local optima of multimodal optimisation problems (or both) by using considerably higher mutation rates than those that have traditionally been recommended [2, 6, 7, 14, 34, 36]. In contrast to generational evolutionary systems where all generated solutions form the new population, the use of artificially introduced elitism in steady-state EAs and GAs allows them to considerably increase the mutation rate without quickly deteriorating their performance i.e., it does not matter if solutions of low quality are created by the higher mutation rates as these will not be accepted [5]. This advantage has been exploited recently by so called *Fast EAs* that use power-law mutation operators which allow for large mutations more often than the binomial distributions of the traditional standard bit mutations (SBMs) of EAs [19, 22, 23]. However, the elitism introduced into steady-state algorithms does not allow them to be competitive on problems where non-elitist algorithms (e.g., simulated annealing) are particularly efficient [13, 24, 29, 32].

Artificial immune systems (AISs), inspired by Burnet's clonal selection working principle of the natural immune system of vertebrates [3], naturally apply high mutation rates (i.e., hypermutation) which have also been proven to escape from local optima more efficiently than SBMs. Furthermore, by using an ageing operator, they have been shown to also be capable of escaping from local optima by accepting solutions of lower quality (i.e., essentially through non-elitism). The effectiveness of both hypermutations and ageing at escaping local optima has been proven on standard multimodal benchmark functions with significant structures from the literature [10], as well as for identifying arbitrarily good approximations for NP-Hard problems such as NumberPartitioning (EAs using SBMs may get stuck on poor 4/3 approximations) [9, 37]. Traditional AISs, such as Opt-IA [12] and B-Cell [25], exhibit this greater exploration capabilities at the expense of being slower during the exploitation phases of the optimisation (i.e., hillclimbing). Recently, though, so called *Fast AISs* have been presented that are asymptotically as fast as traditional randomized local search (RLS) and EAs at hillclimbing, while still outperforming them during the exploration phases [8]. Such improved performance during the hillclimbing

FOGA '21, September 6–8, 2021, Virtual Event, Austria

ACM ISBN 978-1-4503-8352-3/21/09…$15.00
https://doi.org/10.1145/3450218.3477305

phases even allows for linear speed-ups of the AISs to approximate Number Partitioning [11].

A major problem with the generality of the *Fast AISs* is that the hypermutation parameter should be set such that the mutation rate should be *high* for optimal performance at escaping local optima by performing large mutations, and *low* for doing so by accepting lower quality solutions by using the non-elitist capabilities of the ageing operator. In particular, if the mutation rate is too high in the latter case, once a local optimum is escaped from, then there is a high probability that the algorithm returns to it via a large mutation if its basin of attraction is large. Hence, on one hand, setting the parameter to appropriate values requires considerable problem knowledge. On the other hand, a fixed parameter value inevitably leads to suboptimal performance for problems where both the elitist and non-elitist strategies are required at different moments.

In this paper we propose an *Adaptive* Fast AIS that changes the hypermutation rate automatically to identify whether it is more effective to escape from local optima via large mutations or by accepting solutions of lower quality. The adaptive mechanism we propose to use is inspired by the standard 1/5 rule traditionally used in evolutionary computation [1] but differs considerably from it by increasing the mutation rate with the decrease of the success rate, rather than the other way round. The insight for this considerable change is that during the hillclimbing phases where improvements are easier to identify, low mutation rates suffice, while on local optima where improving solutions are harder to identify, then higher mutation rates should be helpful. We rigorously prove that the *Adaptive* Fast AIS either outperforms or performs just as well as the recently introduced Fast EAs and Fast AISs on the same multimodal benchmark functions where the latter were shown to be beneficial. Furthermore we introduce a more general and harder multimodal benchmark function where it is necessary to adapt the hypermutation rate because we show that the state-of-the-art *fast* algorithms with static rates are at least super-polynomially slower.

2 PRELIMINARIES

In this section, we first introduce the artificial immune system framework that we will use throughout the paper, the power law mutation operators (i.e., hypermutations) used by the Fast EAs and Fast AISs and our proposed adaptive Fast AIS. Afterwards we describe the standard benchmark functions from the literature that we will use to assess the performance of the algorithms.

2.1 Algorithms from the Literature

A bare-bones (1+1) AIS is shown in Algorithm 1 [10, 12]. This simple algorithmic framework will suffice to show the benefits of adapting the hypermutation rate for multimodal optimisation. The algorithm uses both characteristic operators of an AIS: ageing and hypermutation. The AIS applies *hybrid* ageing which is the best known for escaping local optima [9, 10, 33]. The algorithm is initialised by choosing a bit-string representing a candidate solution (called *b-cell* or equivalently *individual*) uniformly at random with an assigned age of zero. At each iteration of the *while loop* an offspring is created by performing a *hypermutation*. If the offspring improves the fitness of its parent, its age will be set to zero. Otherwise, it inherits the age

Algorithm 1 (1+1) AIS

1: Initialise $x \in \{0,1\}^n$ uniformly at random and set $x.age := 0$;
2: evaluate $f(x)$;
3: **while** the termination condition is not satisfied **do**
4: $x.age := x.age + 1$;
5: $y := Hypermutation(x)$;
6: evaluate $f(y)$;
7: **if** $f(y) > f(x)$ **then**
8: $y.age := 0$;
9: **else**
10: $y.age := x.age$;
11: **for** $w \in \{x, y\}$ **do**
12: **if** $w.age \geq \tau$ **then**
13: with probability $1/2$, reinitialise w uniformly at random with $w.age = 0$;
14: Set $x = \arg\max_{z \in \{x,y\}} f(z)$;

of the parent. In the next step, if any of these b-cells' age exceeds a threshold τ, that b-cell will be reinitialised with probability $1/2$ and its age will be set to zero. Finally, the best b-cell will be chosen to evolve in the next generation.

For the hypermutation variation step we will consider the recently proposed power-law operators used by the Fast EA and the Fast AIS that have been shown to escape local optima more efficiently than traditional EAs. The Fast EA uses a heavy-tailed power-law distribution [19]. It flips each bit in the bit-string representing the parent solution with probability α/n where the *mutation rate* α is chosen in each generation according to the following probability distribution (which for future reference will be used by the *Adaptive Fast (1+1) AIS*$_\beta$):

$$p(\alpha) := \frac{\alpha^{-\beta}}{\sum_{i=1}^{n} i^{-\beta}}. \tag{1}$$

Here the parameter β was defined to be strictly larger than 1, and a parameter value of $\beta = 1.5$ was recommended when the operator was originally introduced. Compared to the strongly concentrated binomial distribution of SBM (each bit is flipped in every generation with a static probability α/n - usually $\alpha = 1$), this power-law distribution allows for a greater balance between large and small mutations. We make two minor modifications to the operator: 1) rather than flipping each bit with probability α/n, the operator will flip *exactly* α bits; 2) we extend the tail of the distribution to choose α within $[1, \cdots, n]$ rather than within $[1, \cdots, \frac{n}{2}]$ as originally proposed (i.e., the sum $\sum_{i=1}^{n/2} i^{-\beta}$ has been changed to $\sum_{i=1}^{n} i^{-\beta}$ in the denominator of (1)). The former change simplifies the analysis without affecting the average performance of the operator considerably. The latter modification has recently been considered by [22]. Both changes allow to make fairer comparisons with the hypermutation operator used by the Fast AIS which can flip any exact number of chosen bits. When this operator is used in line 5 of Algorithm 1, we call the algorithm Fast (1+1) AIS$_\beta$.

The hypermutation operator of the originally proposed Fast AIS uses a similar distribution [8]. However, the distribution is symmetric i.e., the probability of flipping k bits is the same as that

of flipping $n-k$ bits for all $k \in \{0,\dots,n\}$. Thus, while the expected number of bit-flips is always $n/2$, the β parameter determines the shape of the distribution. Another distinction is that, instead of choosing a mutation rate α and flipping each bit with probability α/n, the Fast AIS uses the power-law distribution to pick *mutation potentials* - the number of bits to flip (the reason why we made the first modification to the Fast EA operator for a fairer comparison of the distributions). In particular, the operator picks the mutation potential $M \in \{0,1,\dots,n\}$ with probability $p(M)$ where $p(M) : \{0,1,\dots,n\} \to (0,1]$, and then flips exactly M bit-positions picked uniformly at random without replacement (this distribution for future reference will be used by the *Adaptive Fast (1+1) AIS$_{s\beta}$*):

$$p(M) := \frac{(\max\{\min\{M, n-M\}, 1\})^{-\beta}}{\sum_{k=0}^{n}(\max\{\min\{k, n-k\}, 1\})^{-\beta}}, \tag{2}$$

We include 0 in the support of M here since it is essential to flip 0-bits for the efficiency of population based evolutionary algorithms, in particular as it impacts the take-over times of the population [5–7, 38]. However, in trajectory-based algorithms such as the ones discussed in this paper, we recommend to exclude not flipping any bits as a possible outcome. Hence, for the rest of the paper we will assume that $M \in \{1,\dots,n\}$. The originally introduced Fast AIS had an equivalent distribution of mutation sizes for each fitness evaluation to (2) when $\beta = 1$ [8]. However, any $\beta \geq 1$ may be applied. It should be pointed out that the originally proposed operator flipped n bits at every operation one by one and used a probability distribution very similar to (2) to determine whether to evaluate the solution after the M-th bit-flip and stopped the hypermutation as soon as an improvement was detected (i.e., *stop at first constructive mutation*). However, this is not necessary as shown in [11] and for a fairer comparison with the distribution of the Fast (1+1) AIS$_\beta$ we remove this feature and only evaluate the solution once all the M bits are flipped. If this hypermutation operator is used in the framework of Algorithm 1, we call the resulting algorithm Fast (1+1) AIS$_{s\beta}$. Fig. 1 compares the two power-law distributions with that of SBM.

Both algorithms exhibit their best performance at escaping local optima via large mutations when the parameter β is set to a value close to 1. However, with such a parameter value, the operators lose effectiveness in conjunction with the ageing operator for escaping from local optima with large basins of attraction by accepting solutions of inferior quality, because the high mutation rates lead the algorithm back to the basin of attraction of the original local optimum with high probability [11].

2.2 Adaptive Mechanism

The mechanism we propose to use to adapt the mutation rate in the (1+1) AIS is inspired by the 1/5 rule traditionally used in evolutionary computation for continuous optimisation [26]. The method has also been applied successfully in discrete optimisation to automatically adapt the offspring population size of crossover-based algorithms [16, 17] and the duration of the learning period in online algorithm selection (i.e., hyper-heuristics) [20, 30]. While commonly used in continuous optimisation, the 1/5 rule adaptation has rarely been rigorously studied to automatically adapt the mutation rate in combinatorial optimisation, with probably the only exception

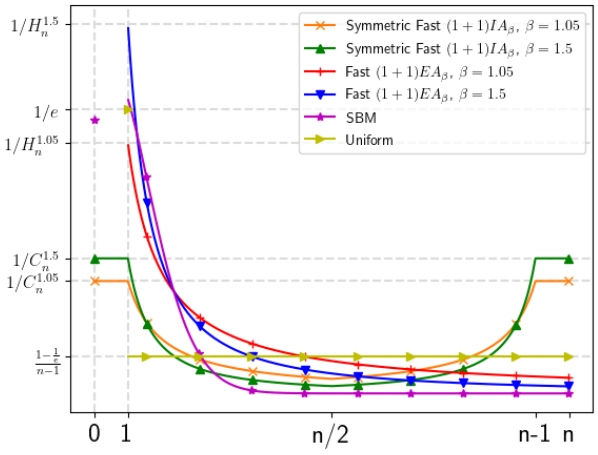

Figure 1: The probability of flipping exactly k bits for the extended heavy-tailed mutation operator of Fast (1+1) AIS$_\beta$ (red and blue) and the symmetric heavy tailed mutation operator of Fast (1+1) AIS$_{s\beta}$ (green and orange) for different β values. The SBM used by standard EAs (purple) and the uniform heavy tailed mutation of Fast (1+1) EA$_{\text{UNIF}}$ [23] with $p = 1/e$ (yellow) are added for comparison. H_n^β and C_n^β denote the denominators of (1) and (2) respectively. The input size is set to $n = 14$ for visualisation.

of [18] that applied it to adapt the mutation rate of the (1+1) EA for the unimodal Leading Ones benchmark function. In the context of mutation rate adaptation, the traditional 1/5 rule increases the mutation rate by a multiplicative factor F if an improving mutation is performed, while it decreases it by a smaller factor (usually $F^{1/4}$) every time a mutation fails to identify an improvement. The mechanism we propose is additionally novel because it considerably differs from the traditional 1/5 rule in two ways. Firstly, with the aim of having high mutation rates when stuck for a long time on a local optimum, the mutation rate is decreased with a success and increased with a failure (the opposite of how the traditional mechanism works). Secondly, since when a local optimum is escaped, our aim is to hillclimb the newly identified gradient, we set the mutation rate to its minimum straight away after an improvement. Formally, assuming maximisation, let x, y, $\beta_{\min}$ and $\beta_{\max}$ be respectively the parent, the offspring, and the bounds on the minimum and maximum values that the parameter β may take, and β_t the value at time step t, then the parameter value at time step $t+1$ will be (update mechanism):

$$\beta_{t+1} = \begin{cases} \max\left\{\beta_t \cdot \left(1 - \frac{\ln\left(\frac{\beta_{\max}}{\beta_{\min}}\right)}{\rho}\right), \beta_{\min}\right\} & \text{if } f(y) \leq f(x), \\ \beta_{\max} & \text{otherwise.} \end{cases}$$

Here the parameter ρ is the number of failures it takes to decrease β from its maximum value $\beta_{\max}$ to its minimum value $\beta_{\min}$. For the analysis we pick $\beta_{\min} \in \{1, 1+\epsilon\}$ for some constant $\epsilon > 0$ in order to have the highest probability of making large jumps, while maintaining a probability of $\Omega(1/\log n)$ and $\Omega(1)$ respectively for local moves. We set $\beta_{\max} = (\log n)^2$ and $\rho = \Omega(n \log n)$ so that for the first $\rho/2$ iterations after an improvement, the algorithms have:

$$\beta \geq \beta_{\max} \cdot \left(1 - \frac{\ln\left(\frac{\beta_{\max}}{\beta_{\min}}\right)}{\rho}\right)^{\rho/2} \geq \beta_{\max} \cdot e^{-\frac{1}{2}\cdot \ln\left(\frac{\beta_{\max}}{\beta_{\min}}\right)}$$
$$\geq \beta_{\max} \cdot \left(\frac{\beta_{\min}}{\beta_{\max}}\right)^{1/2} \geq \log n$$

Thus, with high probability the hypermutation can discover its entire Hamming neighbourhood in $O(n \log n)$ iterations before sampling any solution outside of it, which has a polynomially small probability while the current $\beta = \Omega(\log n)$.

If we use the mechanism to adapt the hypermutation operator of (1) we call the resulting algorithm *Adaptive Fast (1+1) AIS$_\beta$* , while if it is adapting the rate for the symmetric distribution of (2), we call the algorithm *Adaptive Fast (1+1) AIS$_{s\beta}$*.

Recently a stagnation detection mechanism was introduced in the literature for steady-state EAs to identify whether they are on a local optima, and consequently increase the mutation rate [35]. Apart from the different motivations behind our work and theirs, one crucial difference between the update mechanisms, is that stagnation detection only increases the standard bit mutation rate above the standard 1/n rate, when with high probability no neighbouring improvements exist, while our proposed mechanism increases the rate after each unsuccessful hypermutation (at a rate than depends on parameter ρ). Another difference is that the former flips 1 bit in expectation until stagnation is detected, while the aim behind our proposed mechanism is to always flip a linear number of bits in expectation as inspired by the somatic hypermutations occurring in the immune system (i.e., the expected number of bits that flip does not change, just the distribution defined by the parameter β which we adapt).

Lemma 1. *The probability that the Adaptive Fast (1+1) AIS$_\beta$ and the Adaptive Fast (1+1) AIS$_{s\beta}$ with $\beta \in [1, (\log n)^2]$ flip exactly one bit at a particular hypermutation operation is at least,*

- $(\ln n + O(1))^{-1}$ *in general and* $1 - o(1)$ *when* $\beta > \log n$ *for the Adaptive Fast (1+1) AIS$_\beta$,*
- $(2 \cdot \ln n)^{-1}$ *in general and* $\frac{1}{3} - o(1)$ *when* $\beta > \log n$ *for the Adaptive Fast (1+1) AIS$_{s\beta}$.*

Proof. For the probability of flipping 1-bit we refer to (1) and (2) and note that in both equations we have the numerator equal to 1. We will now bound the normalization factors in the denominators separately. We pessimistically assume $\beta = 1$. Thus, for Eq.(1) we have:

$$\sum_{i=1}^{n} i^{-1} \leq \ln n + O(1)$$

and for Eq. (2), we have (recall that we normalize over $M \in \{1, \ldots, n\}$ since we avoid flipping 0 bits due to not having a population):

$$\sum_{k=1}^{n} (\max\{\min\{k, n-k\}, 1\})^{-\beta} =$$
$$1 + \sum_{k=1}^{\lfloor n/2 \rfloor} k^{-\beta} + \sum_{k=\lceil n/2 \rceil}^{n-1} (n-k)^{-\beta}$$
$$\leq \ln \frac{n}{2} + \ln \frac{n}{2} + 1 \leq 2 \ln n$$

When $\beta > \log n$, the denominator for Eq. (1) is $1 + o(1)$ since for any $i \geq 2$, $i^{-\beta} = O(n^{-1})$ and if we exclude $i = 2$, for $i > 2$, we have $i^{-\beta} = o(n^{-1})$. Thus, we can bound the denominator from above by $1 + o(1)$.

On the other hand, we have three terms ($k \in \{1, n-1, n\}$) in the denominator of Eq. (2) which are equal to $1^{-\beta} = 1$, while we can bound the remaining terms as we did for Eq. (1). The result is $3 + o(1)$. □

2.3 Benchmark Functions

We will evaluate the performance of the adaptive algorithms against the static ones on a range of widely used pseudo-Boolean benchmark function classes $f : \{0,1\}^n \to \mathbb{R}$. These have been designed to reflect significant aspects of optimisation problems that are expected to appear in real-world optimisation applications. Without loss of generality, we will consider the instances of each function class that have the global optimum in the 1^n bit-string since the behaviour of the algorithms is the same for all function class instances.

2.3.1 Unimodal Functions. We will start our analysis by establishing the hillclimbing performance of the adaptive algorithms. For this purpose standard unimodal functions from the literature will be used.

The aim of the well-studied OneMax function is to identify a hidden bit-string by minimizing the Hamming distance to it of the candidate solutions. To this end, the function returns the number of correctly identified bit-positions (i.e., the number of ones when the 1^n bit-string is used as target). The function reflects the typical characteristic of optimisation problems, that the closer the algorithm gets to the optimum, the harder it is to identify improving solutions. When the target is the 1^n bit-string, it is defined formally as $\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i$ (see Fig. 2). The unary unbiased black box complexity of OneMax is $\Theta(n \log n)$ [28] meaning that no algorithm using an unbiased mutation operator may be faster. Both the Fast (1+1) AIS$_\beta$ and the Fast (1+1) AIS$_{s\beta}$ optimise the function in this best possible expected asymptotic runtime [8, 19].

LeadingOnes is a slightly harder unimodal function. The aim of the problem is that of identifying a hidden permutation of the bit-string. If the 1^n bit-string is used as global optimum, then the function returns the number of consecutive 1-bits before the first 0-bit: $\textsc{LeadingOnes}(x) := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$. Its unbiased black box complexity is $\Theta(n^2)$ which is also met by the two power-law mutation operators [8, 19].

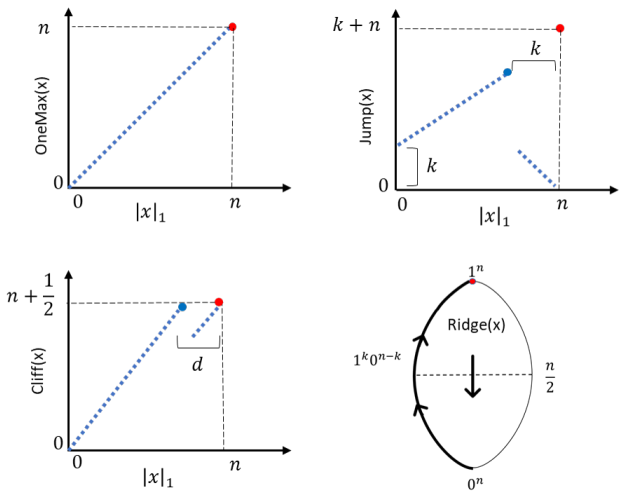

Figure 2: Benchmark functions. $|x|_1$ shows the number of 1-bits in a bit-string. The Ridge function is illustrated over the Boolean hypercube $\{0,1\}^n$.

The Ridge function class is slightly harder than both OneMax and LeadingOnes. First a OneMax slope needs to be followed to the 0^n bit-string. Afterwards, the 0-bits have to be flipped in consecutive order without touching the tail of the bit-string (i.e., only bit-strings of consecutive 1-bits followed by consecutive 0-bits are on the ridge of higher fitness). While the OneMax slope allows to assess algorithmic performance as improvements become harder with progress, the Ridge slope is used to assess performance when improvement probabilities do not change with the position on the slope:

$$\textsc{Ridge}\ (x) := \begin{cases} (k+1)\cdot n & \text{if } x = 1^k 0^{n-k}, k \in \{1,\dots,n\} \\ n - |x|_1 & \text{otherwise.} \end{cases}$$

Unary unbiased search heuristics typically require $\Theta(n^2)$ expected function evaluations, a runtime that is met by the power-law hypermutation operators. We will use Ridge as a basis for the design of a multimodal benchmark function where it is necessary to adapt the mutation rate for the AISs to run in polynomial expected runtimes. Using Ridge rather than OneMax for the hillclimbing phases will allow us to make the function more challenging to optimise.

2.3.2 Multimodal Functions. Multimodal functions are generally used to test the exploration abilities of search heuristics including their effectiveness at escaping local optima. We consider two different classes of multimodal benchmark functions which allow performance assessment in different scenarios that are expected to be typically encountered in real-world optimisation.

The $\textsc{Jump}_k$ function class is widely used in the performance analysis of randomized search heuristics. It consists of two consecutive OneMax slopes of length $n-k$ and k, respectively. The first slope leads the algorithm towards the local optima and the second OneMax slope leads back to it (see Fig. 2). Since the global optimum is the only search point of better quality than the local optima, elitist algorithms typically have to perform a large mutation by flipping simultaneously k exact bits to optimise the function. In particular, for elitist algorithms, the function models the hardest-to-escape local optima with basin of attraction of size k, since there exists only one search point at Hamming distance k from the local optimum, that has better fitness. Due to the second slope leading back to the local optima, the function is also very hard for non-elitist algorithms that may accept solutions of inferior quality on the second slope. The function is formally defined as:

$$\textsc{Jump}_k(x) := \begin{cases} k + \sum_{i=1}^n x_i & \text{if } \sum_{i=1}^n x_i \le n-k \\ & \text{or } \sum_{i=1}^n x_i = n, \\ n - \sum_{i=1}^n x_i & \text{otherwise.} \end{cases}$$

$\textsc{Jump}_k$ was the function class used to show the effectiveness of the power-law hypermutation operators over the traditional SBM operators [8, 19]. In particular, the Fast (1+1) AIS_β and the Fast (1+1) $\text{AIS}_{s\beta}$ have respectively expected runtimes of $O(k^\beta \binom{n}{k})$ and $O((\min\{k, n-k\})^\beta \binom{n}{k})$ (if $\beta > 1$, and $O((\min\{k, n-k\})^\beta \log n \binom{n}{k})$ if $\beta = 1$) which are exponentially smaller than the $\Theta(n^k)$ expected runtime required by SBM [21]. From the runtime bounds it can be appreciated why the parameter β should be set arbitrarily close to 1 for the operators to have their best possible performance for $\textsc{Jump}_k$. A value of $\beta = 1$ is the best possible if more than n^ϵ bits, for any arbitrarily small constant $\epsilon > 0$, have to be flipped to escape from the local optima. We point out that the best possible expected runtime achievable with SBM is $O((en/k)^k$ which is matched by the stagnation detection adaptive (1+1) EA [35].

The $\textsc{Cliff}_d$ function class is often used to assess the performance of non-elitist algorithms. It was originally designed as an example problem where non-elitist EAs outperform elitist ones [24]. This function includes a OneMax slope with length $n-d$ which leads the algorithms towards the local optima. The local optima are followed by a second OneMax slope (of length d) of lower fitness which leads to the global optimum (see Fig. 2). While for elitist algorithms the function is just as hard as $\textsc{Jump}_k$, non-elitist algorithms, by accepting inferior solutions, can easily find the second slope and then hillclimb up to the global optimum. $\textsc{Cliff}_d$ is defined as:

$$\textsc{Cliff}_d(x) = \begin{cases} \sum_{i=1}^n x_i & \text{if } \sum_{i=1}^n x_i \le n-d, \\ \sum_{i=1}^n x_i - d + 1/2 & \text{otherwise.} \end{cases}$$

Some non-elitist algorithms such as Metropolis still require exponential time in the length of the second slope [29]. When the second slope has linear length (i.e., the function is hardest for elitist algorithms) (1+1) EAs and randomized local search algorithms equipped with ageing are known to optimise the function in $O(n \log n)$ function evaluations i.e., the best runtime achievable by unary unbiased search heuristic on any function with unique optimum [28], which is also met by the non-elitisit Move-Acceptance hyper-heuristic [29]. With a small β parameter value close to 1 (optimal for $\textsc{Jump}_k$), the Fast (1+1) AIS_β and Fast (1+1) $\text{AIS}_{s\beta}$ will be inefficient because the high mutation rates will cause search points on the second slope to return to the local optima with high probability. The expected runtime decreases with higher values of β and will reduce to $O(n \log n)$ in expectation for parameter values in the order of $\beta = \Omega(\log n)$ because the mutation operators flip one bit most of the time with

such a setting. We will show that by automatically adapting the hypermutation rate, optimal performance can be achieved for both $\textsc{Cliff}_d$ and $\textsc{Jump}_k$.

3 UNIMODAL FUNCTION ANALYSIS

In this section we will analyse the hill-climbing capabilities of the Adaptive Fast (1+1) AIS$_\beta$ and the Adaptive Fast (1+1) AIS$_{s\beta}$ on unimodal functions. The essence of the following result is that we can guarantee that an improvement will be achieved before β drops to levels that yield super-constant probabilities for single bit flips. A constant probability for single bit flips allows the algorithm to be asymptotically optimal when hill-climbing.

Lemma 2. *Let $f : \{0,1\} \to \mathbb{R}$ be a unimodal function, F be the number of distinct values f can take and $E[T]$ be the the expected runtime on f of the Adaptive Fast (1+1) AIS$_\beta$ (Adaptive Fast (1+1) AIS$_{s\beta}$) with $\tau \geq \rho/2 \geq (6+\epsilon)\cdot n \cdot \ln(\max(F,n))$ for some arbitrarily small $\epsilon > 0$ and $\beta \in [1, (\log n)^2]$. Then, $E[T] \leq O(n \log n) + E[T|\mathcal{E}_s]$ where $\mathcal{E}_s$ is the event that the optimum is found before a reinitialisation.*

Proof. We will define three events that can happen after the algorithm is initialised with a solution picked uniformly at random.

- $\mathcal{E}_s$: The optimum is found before reinitialisation.
- $\mathcal{E}_d$: The algorithm reinitialises before any improvement.
- $\mathcal{E}_r$: The algorithm reinitialises after at least one improvement.

We will denote the probabilities of these events as p_s, p_d, p_r respectively. Since the events are mutually exclusive and exhaustive, $p_s + p_d + p_r = 1$. Hence, the law of total expectation for the expected runtime $E[T]$ implies:

$$E[T] = E[T|\mathcal{E}_s]\cdot p_s + E[T|\mathcal{E}_d]\cdot p_d + E[T|\mathcal{E}_r]\cdot p_r \tag{3}$$

We bound the conditional expectations as follows,

- $E[T|\mathcal{E}_d] \leq E[T] + \tau + 4$, since after τ iterations without improvement the probability that both the parent and the offspring to die due to ageing is $1/4$ and the expected time until it happens is 4. After the reinitialisation the expected time is $E[T]$ by definition.
- $E[T|\mathcal{E}_r] \leq E[T] + F\cdot\tau + 4\cdot F$, since the argument for the previous case can be repeated. The algorithm can spend at most $\tau+4$ generations in expectation before improving upon each distinct fitness value.

We bound the probabilities as follows,

- $p_d \leq e^{-\frac{\tau}{2\cdot n\ln n}}$: If the algorithm picks mutation size 1, which happens with probability at least $1/(2\cdot\ln n)$, then the conditional probability of improvement is at least $1/n$ since both operators are unbiased and f is unimodal. For the ageing operator to trigger, the algorithm should sample at least τ offspring which do not improve upon their parent. The probability of this event can be bounded above by:

$$p_d \leq \left(1 - \frac{1}{2\cdot n\ln n}\right)^{\tau} \leq e^{-\frac{\tau}{2\cdot n\ln n}} \tag{4}$$

For $\tau \geq (6+\epsilon)n\ln n$, for any constant $\epsilon > 0$

$$p_d \leq e^{-3} \tag{5}$$

- $p_r \leq F\cdot e^{-\frac{\rho}{6\cdot n}+\frac{\rho}{4\cdot n\ln n}}\cdot e^{-\frac{\tau}{2\cdot n\ln n}}$: If the algorithm improves at least once, then in order to reinitialise it is necessary that for at least $\rho/2$ generations the algorithm must fail to improve while the β parameter stays above $\log n$. When $\beta > \log n$, the probability of the operator flipping one bit position is at least $1/3 - o(1)$ due to Lemma 1. If $\tau > \rho/2$, then the algorithm must also fail to improve in the following $\tau - \rho/2$ iterations where the improvement probability is at least $1/(2\cdot n\ln n)$ as in the previous case. Thus, using a union bound over all possible fitness values:

$$p_r \leq F\cdot\left(1-\frac{1}{3n}\right)^{\rho/2}\cdot\left(1-\frac{1}{2\cdot n\ln n}\right)^{\tau-\rho/2} \leq F\cdot e^{-\frac{\rho}{6\cdot n}-\frac{\tau-\rho/2}{2\cdot n\ln n}}$$
$$\leq F\cdot e^{-\frac{\rho}{6\cdot n}+\frac{\rho}{4\cdot n\ln n}}\cdot e^{-\frac{\tau}{2\cdot n\ln n}}$$

- For $\tau \geq \frac{\rho}{2} \geq (6+\epsilon)n\ln F$ for some constant $c > 0$ and for an arbitrarily small $\epsilon > 0$:

$$e^{-\frac{\rho}{6\cdot n}+\frac{\rho}{4\cdot n\ln n}} \leq F^{-2}. \tag{6}$$

- For $\tau \geq \frac{\rho}{2} \geq (6+\epsilon)\cdot n\ln F$, $p_s = \Omega(1)$: This result follows from $p_s = 1 - p_d - p_r$ and equations (6) and (5).

Thus, from (3) we obtain:

$$\begin{aligned} E[T] &\leq E[T|\mathcal{E}_s]\cdot p_s + (E[T]+\tau+O(1))\cdot p_d \\ &\quad + (E[T]+F\cdot\tau+O(F))\cdot p_r \\ (1-p_d-p_r)E[T] &\leq E[T|\mathcal{E}_s]\cdot p_s + (\tau+O(1))\cdot p_d \\ &\quad + (F\cdot\tau+O(F))\cdot p_r \\ E[T] &\leq E[T|\mathcal{E}_s] + \frac{\tau\cdot p_d + F\cdot\tau\cdot p_r}{\Omega(1)} \end{aligned}$$

We will now bound $\tau\cdot p_d + F\cdot\tau\cdot p_r$ to obtain our claim,

- $\tau\cdot p_d = O(n\log n)$: Due to (4), $\tau\cdot p_d < e^{-\frac{\tau}{2\cdot n\ln n}}\cdot\tau$. The right hand side decreases for all $\tau \geq 2\cdot n\ln n$.
- $F\cdot\tau\cdot p_r = O(n\log n)$: Since $p_r \leq F\cdot e^{-\frac{\rho}{6\cdot n}+\frac{\rho}{4\cdot n\ln n}}\cdot e^{-\frac{\tau}{2\cdot n\ln n}}$,

$$\begin{aligned} F\cdot\tau\cdot p_r &\leq \left(e^{-\frac{\rho}{6\cdot n}+\frac{\rho}{4\cdot n\ln n}}\cdot F^2\right)\cdot e^{-\frac{\tau}{2\cdot n\ln n}}\cdot\tau \\ &\leq O(1)\cdot O(n\log n) \end{aligned}$$

where the $O(1)$ is due to eq (6) and $O(n\log n)$ is obtained as in the previous item.

□

Theorem 3. *The Adaptive Fast (1+1) AIS$_\beta$ and the Adaptive Fast (1+1) AIS$_{s\beta}$ with the interval $\beta \in [1,(\log n)^2]$ and parameters $\tau \geq \rho/2 \geq (6+\epsilon)\cdot n\log n$ for an arbitrarily small constant $\epsilon > 0$, optimise $\textsc{OneMax}$ in $\Theta(n\log n)$, $\textsc{LeadingOnes}$ in $\Theta(n^2)$ and $\textsc{Ridge}$ in $O(n^2)$ expected fitness function evaluations.*

Proof. The number of distinct values for our functions of interest are at most $2n$ (which is the case for the $\textsc{Ridge}$ function). Thus, our stated τ and ρ parameters satisfy the scope of Lemma 2. Since all our claimed upper bounds are $\Omega(n\log n)$, we will only find upper bounds on the expected time given that ageing does not trigger.

For any unimodal function f, the probability that an offspring has a better fitness than its parent is at least Pr (single bit is flipped) $/n$

since in any unimodal function at least one Hamming neighbour has a better fitness value than any current suboptimal solution. Using that all our functions are unimodal the expected time until the first improvement occurs is at most $O(n \log n)$ since both algorithms flip a single bit with probability at least $\Omega(1/\log n)$. We will next bound the expected time after the first improvement occurs.

Let $E[T_i]$ denote how many expected hypermutation operations it takes to improve the fitness-function value from i to any value greater than i. Moreover, let $\mathcal{E}_i$ denote the event that the β value for the algorithm stays above $\log n$ while the current fitness is i. By the law of total expectation we divide $E[T_i]$ into two additive terms:

$$E[T_i] = \Pr(\mathcal{E}_i) \cdot E[T_i|\mathcal{E}_i] + \Pr(\neg\mathcal{E}_i) \cdot E[T_i|\neg\mathcal{E}_i]. \tag{7}$$

Since β is set to $(\log n)^2$ after an improvement, the event $\mathcal{E}_i$ occurs if and only if the algorithm fails to improve $(6+\epsilon) \cdot n \ln n \leq \rho/2$ iterations consecutively as in $\rho/2$ generations β multiplicatively decreases from $(\log n)^2$ to $\log n$. As the β value stays above $\log n$ until we fail $(6+\epsilon) \cdot n \ln n$ times after the last improvement, we can bound $\Pr(\neg\mathcal{E}_i)$ from above:

$$\begin{aligned}\Pr(\neg\mathcal{E}_i) &\leq \left(1 - \frac{\Pr(\text{single bit is flipped})}{n}\right)^{(6+\epsilon)\cdot n \ln n} \\ &= O(n^{-2}),\end{aligned}$$

since $\Pr(\text{single bit is flipped}) = \frac{1}{3} - o(1)$ for all $\beta > \log n$.

Since for all $\beta \in [1, (\log n)^2]$ we have $\Pr(\text{single bit is flipped}) = O(1/\log n)$, the conditional expectation $E[T_i|\neg\mathcal{E}_i] = O(n \log n)$. Thus, we can bound the second term in (7) by $O(\log n/n)$.

When calculating $E[T_i|\mathcal{E}_i]$ for the functions LeadingOnes and Ridge , we can use the upper bound on the improvement probability $\Pr(\text{single bit is flipped})/n$ and conclude that $E[T_i|\mathcal{E}_i] = O(n)$ since $\mathcal{E}_i$ implies $\Pr(\text{single bit is flipped}) = \Omega(1)$. In order to find the total expected time of $O(n^2)$ we recall that both LeadingOnes and Ridge have $O(n)$ different fitness values.

For the OneMax function, $E[T_i|\mathcal{E}_i]$ can be smaller than $O(n)$ depending on i. In particular, it is sufficient that the hypermutation operator flips a single 0-bit into a 1-bit and does not flip any other bits. This mutation occurs with probability $\Pr(\text{single bit is flipped}) \cdot \frac{i}{n} = \frac{c \cdot i}{n}$ for some constant $c > 0$ and its expected waiting time is at most $E[T_i|\mathcal{E}_i] \leq n/(c \cdot i)$. When summed over different values for $i \in \{1, \ldots, n\}$, the total waiting time $\sum_{i=1}^{n-1} \frac{n}{c \cdot i}$ is in the order of $O(n \log n)$. Thus, for all considered functions the second term in (7) is a small order term, and the claimed upper bounds follow.

The matching lower bounds are due to the unary unbiased black-box complexity of the functions [28], which asymptotically match the upper bounds for OneMax and LeadingOnes functions. □

4 MULTIMODAL FUNCTIONS WHERE ADAPTATION IS ADVANTAGEOUS

We will now focus on the two multi-modal benchmark functions that require different β values to be optimised effectively. Our adaptive mechanism allows the β value to drop to its minimum when on the local optima of either Jump or Cliff. While high mutation rates (i.e., low β values) are necessary to solve the Jump function effectively, for the Cliff counterpart it is critical to flip as few bits as possible to avoid sampling solutions from the first slope. This is easily achieved by the adaptive mechanism which sets $\beta = (\log n)^2$ as soon as an improvement is found on the second slope of the Cliff function.

Theorem 4. *The Adaptive Fast (1+1) AIS$_\beta$ and the Adaptive Fast (1+1) AIS$_{s\beta}$ with the interval $\beta \in [1, (\log n)^2]$ and $\tau \geq \rho/2 \geq (6+\epsilon) \cdot n \log n$ for some constant $\epsilon > 0$, optimise Cliff$_d$ in $O(\frac{\tau \cdot (n \log n)^2}{d^2})$ and Jump$_k$ in $O(k \log n \binom{n}{k})$ expected function evaluations. For the parameter setting $\beta \in [1+\epsilon, (\log n)^2]$ for some arbitrarily small constant $\epsilon > 0$, the expected runtime for Jump$_k$ is at most $O\left(k^{1+\epsilon}\binom{n}{k}\right)$ while the expected runtime for Cliff$_d$ is $O(\frac{\tau \cdot n^2}{d^2})$.*

Proof. We first prove the upper bound for the Jump function. The function is identical to OneMax until one of the local optima (i.e., a solution with $n-d$ 1-bits) is sampled because whether we initialise with fewer than or more than $n-d$ bits we can regardless use $(n - \text{OneMax}(x))/n$ as a lower bound on the probability of improving any solution except the global or local optima. Thus, we use the upper bound from Theorem 1 to conclude that in expected $O(n \log n)$ iterations after any uniformly random initialisation, we sample a local optimum. Pessimistically assuming that the hypermutation fails to sample the optimum (which is the only solution that improves a local optimum) for ρ times after the local optimum is sampled, for the remaining $\tau - \rho \geq \rho/2 = \Omega(n \log n)$ iterations we have $\beta = 1$. With $\beta = 1$ the probability that the hypermutation samples the optimal solution is $\Omega\left((k \log n)^{-1}\binom{n}{k}^{-1}\right)$ since with probability at least $\Omega(1/(k \cdot \log n))$, k bits will be flipped and given that k bits are flipped, the probability that the optimum is sampled is at least $\binom{n}{k}^{-1}$. The expected waiting time until the event occurs is the reciprocal of this probability, $O\left(k \log n \binom{n}{k}\right)$. If the ageing triggers while the current best is at the local optimum, we repeat our argument that in $O(n \log n)$ iterations we reach the local optimum and attempt $\Omega(n \log n)$ times to sample the global optimum, which implies that in expectation it is necessary to climb the OneMax slope $O\left(k \log n \binom{n}{k}\right)/\Omega(n \log n)$ times. Since climbing the OneMax slope takes at most $O(n \log n)$ times, the expected waiting time is still in the order of $O\left(k \log n \binom{n}{k}\right)$. Following the same arguments we can also find that when $\beta \in [1+\epsilon, (\log n)^2]$, the upper bound is $O\left(k^{1+\epsilon}\binom{n}{k}\right)$ since the probability of flipping k bits changes from $\Omega(\frac{1}{k \cdot \log n})$ to $\Omega(k^{-1-\epsilon})$.

For Cliff, the expected time to reach the local optima is the same as above, i.e., $O(n \log n)$. Now, ageing is triggered after τ iterations and then, since $\beta = \beta_{\min}$ due to the time spent on the local optima, with probability $\Omega\left(\frac{d}{n \log n}\right)$ if $\beta_{\min} = 1$ and with probability $\Omega(d/n)$ if $\beta_{\min} = 1+\epsilon$, a solution with more 1-bits is sampled in the same iteration. In the following ageing phase, with constant probability, the parent solution is removed from the population while the offspring is spared. In the following iteration, the survived individual improves again with probability $\Omega\left(\frac{d}{n \log n}\right)$ or $\Omega\left(\frac{d}{n}\right)$ depending on whether $\beta_{\min}$ is equal to 1 or $1+\epsilon$ respectively and sets its age to zero and $\beta = (\log n)^2$. Since each solution with more than $n-d$ 1-bits has at least one Hamming neighbour with better fitness value, following the same arguments of the proof of Theorem 1 we conclude that with probability $1 - O(n^{-2})$, $\beta > \log n$

between two consecutive improvements. Since there are at most n improvements, by the union bound the probability that $\beta > \log n$ until the optimum is found is at least $1 - O(n^{-1})$

Next, we will bound the conditional probability that a solution with at most $n - d$ 1-bits is sampled before the optimum is found given that the current solution has at least $n - d + 2$ 1-bits and that $\beta > \log n$. First, we will bound the probability that a mutation flips a specific number of bit positions. Since $\beta > \log n$, the probability of flipping exactly two positions is $O(2^{-\log n}) = O(1/n)$, three positions is $O(3^{-\log n}) = o(1/n)$, 4 to 8 positions is $O(4 \cdot 4^{-\log n}) = O(1/n^2)$ and more than 8 positions is at most $O(n \cdot 8^{-\log n}) = O(1/n^2)$. Therefore, given that the improvement probability is at least $\Omega(1/n)$ the probability of obtaining a solution with $n - d + 3$ 1-bits before flipping two bits is a constant. Then, the probability of obtaining a solution with $n - d + 4$ 1-bits before flipping three bits is in the order of $1 - o(1)$, the probability of obtaining a solution with $n - d + 8$ 1-bits given that the current solution has at least $n - d + 4$ 1-bits is at least $1 - O(1/n)$. Finally, given that we obtain a solution with $n - d + 8$ 1-bits, the probability of improving before flipping at least 8 bit positions is at least $1 - O(1/n^2)$. Thus, the probability of improving n times before we flip at least 8 bits is at least $1 - O(1/n)$ by the union bound. Combining the failure probabilities, we conclude that with constant probability we sample the optimum before sampling a solution with at most $n - d$ 1-bits. If any of the failures occur we pessimistically assume that the current solution is reinitialised and then the local optima is sampled before the global optimum. Considering that once a local optimum is found and the ageing mechanism is triggered in $O(n \log n) + \tau = O(\tau)$ iterations afterwards, we also find the optimal solution before reinitialisation with probability $\Omega(d^2/(n \cdot \log n)^2)$ ($\Omega(d^2/n^2)$ if $\beta_{\min} = 1 + \epsilon$) in expected $O(n \log n)$ iterations. Thus, the total expected runtime can be bounded above by $O\left(\tau \cdot \frac{(n \log n)^2}{d^2}\right)$ and $O(\tau \cdot \frac{n^2}{d^2})$ respectively. □

5 MULTIMODAL FUNCTIONS WHERE ADAPTATION IS NECESSARY

In this section, we will introduce the CliffWithBasin pseudo-Boolean function where it is necessary to make larger jumps to solutions of lower quality compared to the standard Cliff function. This function will allow us to illustrate how the adaptive hypermutation algorithms are more effective in more challenging multimodal optimisation scenarios where it may be convenient to make large jumps to solutions of lower quality to escape efficiently from local optima. In particular we present a more general multimodal function that they can efficiently optimise while the state-of-the-art hypermutation operators with static parameter β have at least super-polynomially worse performance.

The function is illustrated in Fig. 3 and is defined as:

Definition 1. *For the parameters* $\boldsymbol{a} := (a_1, a_2) \in [n]^2$, $n\left(\frac{1}{2} + \epsilon\right) < a_1 < a_2$ *where* $\epsilon > 0$ *is an arbitrarily small constant, and* $n - a_2 =$

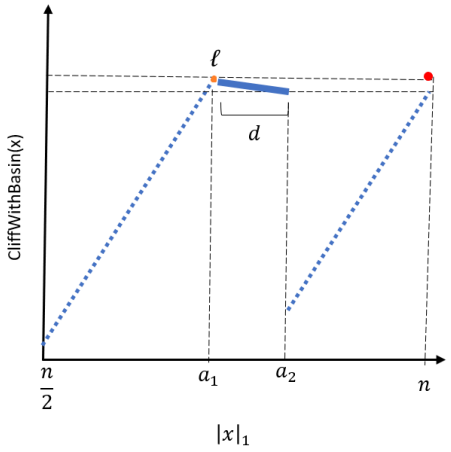

Figure 3: Sketch of the CliffWithBasin function. Blue dashed lines ("Ridge slope") indicate that there exists a unique solution with the plotted function value among solutions with a given $|x|_1$ value. The solid line indicates a multiplicative inverse OneMax slope.

$\Omega(n)$ *let*[1]

$$FS := \left\{x | x = 1^k 0^{n-k} \wedge\ k \in \{\lceil n/2 + 1 \rceil, \ldots, a_1 - 1\}\right\},$$
$$SS := \left\{x | x = 1^k 0^{n-k} \wedge\ k \in \{a_2, \ldots, n\}\right\},$$
$$BA := \{x | a_1 < \textsc{OneMax}\,(x) < a_2\},$$
$$LO := \{x | \textsc{OneMax}\,(x) = a_1\},$$
$$NH := \left\{x | \textsc{OneMax}\,(x) = \left\lceil \frac{n}{2} \right\rceil\right\}.$$

CliffWithBasin$_{\boldsymbol{a}} : \{0,1\}^n \to \mathbb{R}$ *is defined as follows:*

$$\textsc{CliffWithBasin}_{\boldsymbol{a}}(x) := \begin{cases} \textsc{Ridge}\,(x) & \text{if } x \in FS \\ \textsc{Ridge}\,(x) - (n - a_2) + 1/2 & \text{if } x \in SS \\ (a_1 + 1) \cdot n - \left(1 - \frac{1}{\textsc{OneMax}\,(x)}\right) & \text{if } x \in BA \\ (a_1 + 1) \cdot n - \frac{1}{\textsc{LeadingOnes}\,(x)+1} & \text{if } x \in LO \\ (\lceil \frac{n}{2} \rceil + 1) \cdot n - \frac{1}{\textsc{LeadingOnes}\,(x)+1} & \text{if } x \in NH \\ \frac{1}{|\frac{n}{2} - \textsc{OneMax}\,(x)|} & \text{o.w.} \end{cases}$$

The function has two parallel slopes with the second slope having worse fitness as in the Cliff function while it is necessary to flip d exact bit position to move from one slope to the other. Unlike the Cliff function the slopes consist only of Ridge points, which have the form $1^k 0^{n-k}$ for some $k \in \{n/2, \ldots, n\}$. The rest of the search space has gradients that serve to lead the current solution either to the point $1^{n/2}0^{n/2}$ which is the beginning of the first slope or to the local optima $\ell := 1^{a_1}0^{n-a_1}$, i.e., the point with the highest fitness on the first slope. All solutions which have between $a_1 + 1$ and a_2 1-bits (i.e., BA solutions) have smaller fitness values than the local and the global optima but higher than any other solution.

[1]The names of the sets FS, SS, BA, LO, NH abbreviate: first slope, second slope, basin of attraction, leading ones and n-half, respectively.

This basin of attraction forces the algorithms to make a very precise mutation to reach the second slope. Moreover, since the second slope has lower fitness values, even if a solution from the second slope is sampled, it is highly likely that it is mutated into a solution in the basin of attraction, preventing the algorithm from following the second slope to the global optimum. These characteristics make the function particularly challenging for both elitist and non-elitist algorithms. For instance Metropolis would fail to access the second slope of increasing fitness in polynomial expected time due to the large difference in fitness gradient while the Move-Acceptance non-elitist hyper-heuristic would likely require an expected runtime in the order of $O(n^{2d})$ to cross the basin [29].

In the following theorem, we show that constant β values lead to exponential runtime bounds for the CliffWithBasin function. In particular, constant β values cause the hypermutation operator to flip many bits too frequently such that, with overwhelming probability, a solution in the second slope is mutated into a *BA* solution before the global optimum is reached.

Theorem 5. *The Fast (1+1) AIS$_\beta$ and the Fast (1+1) AIS$_{s\beta}$ with $1 \leq \beta = O(1)$ and $\tau > 0$ cannot optimise CliffWithBasin$_{\boldsymbol{a}}$ in fewer than $e^{\Omega(n)}$ expected fitness function evaluations.*

Proof. In order to prove the lower bound on the runtime, we will bound from below the probability of the algorithm reinitialising (i.e., starting a generation with random solution uniformly distributed over the $\{0,1\}^n$ hyperspace) before sampling the optimal solution. We will denote this event as $\mathcal{E}_0$ and use the law of total probability to analyse the conditional probabilities of this event given the age (α_t) and the genotype (x_t) of the current solution at iteration t. Let $\ell := 1^{a_1}0^{n-a_1}$. We will start with the conditional probability of the $\mathcal{E}_0$ given that $x_t = \ell$, and define $\mathcal{E}_1$ as the event of the algorithm sampling the 1^n bit-string in the τ generations following generation t:

$$\begin{aligned}&\Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell)\\ &= \Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell, \mathcal{E}_1) \cdot \Pr(\mathcal{E}_1|\alpha_t = 0, x_t = \ell)\\ &+ \Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell, \neg\mathcal{E}_1) \cdot \Pr(\neg\mathcal{E}_1|\alpha_t = 0, x_t = \ell)\,.\end{aligned}$$

We will next bound the probability that any solution in *LO* is mutated into the 1^n string in a single mutation operation. This mutation requires $n - a_1$ precise bits to be flipped by the operator and its probability is at most $\binom{n}{n-a_1} = n^{-\Omega(n)}$ since even if the mutation operator picks the correct number of bits to be flipped with probability 1, it has to pick each of these $n - a_1$ positions correctly among n alternatives. We can use the union bound to establish that for any $\tau = poly(n)$, $\Pr(\mathcal{E}_1|\alpha_t = 0, x_t = \ell) = n^{-\Omega(n)}$. Since $\Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell, \neg\mathcal{E}_1)$ is equivalent to $\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell)$ we obtain:

$$\begin{aligned}&\Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell)\\ &> (1 - n^{-\Omega(n)}) \cdot \Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell) \qquad (8)\end{aligned}$$

When $\alpha_t \geq \tau$ both the current solution and any offspring solution with equal or less fitness than the parent are each removed from the population with probability 1/2. We first condition on the new offspring having improved the solution which only happens when the global optimum is sampled which has probability $n^{-\Omega(n)}$ and allows us to absorb the conditional probability into the $(1-n^{-\Omega(n)})$ factor. Then, we condition on which individuals are removed from the population. If the parent survives, then in the next generation the conditional probability is equal to $\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell)$ since we repeat the same experiment in the next iteration. If both individuals die, then the event $\mathcal{E}_0$ occurs. Finally, if only the offspring individual survives we denote the resulting event as $\mathcal{E}_s$ and obtain:

$$\begin{aligned}&\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell)\\ &> \frac{1}{4} + \frac{1}{2}\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell)\\ &+ \frac{1}{4} \cdot \Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell, \mathcal{E}_s)\\ &\Longrightarrow\\ &\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell)\\ &> \frac{1}{2} + \frac{1}{2} \cdot \Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell, \mathcal{E}_s)\,.\end{aligned}$$

Next, we will condition on which subspace the survived individual belongs to. For any constant $\epsilon^* < 1/2$, the probability of mutating from any solution with less then a_2 1-bits (which includes all the named sets except *SS*) to any point in *SS* with at least $a_2 + \frac{n^{\epsilon^*/\beta}}{2}$ 1-bits (a *safe solution*) or to the optimal 1^n bit-string is at most $\binom{n}{\frac{n^{\epsilon^*/\beta}}{2}}^{-1} = n^{-\Omega(n^{\epsilon^*/\beta})}$. Conditioning on that this event does not happen, if the survived individual has more than $n(1+\epsilon)/2$ 1-bits and does not belong to *LO*, *BA*, *FS*, *NH*, or *SS* then the uniformly initialised individuals will have a better fitness than the offspring with overwhelmingly high probability and event $\mathcal{E}_0$ will occur. Moreover, all solutions in $\{0,1\}^n \setminus (SS \cup \{1^n\})$ has a sequence of $O(n^2)$ local improvements that lead them to ℓ while the probabilities for the mutation to sample a safe solution or the optimum is at most $n^{-\Omega(n^{\epsilon^*/\beta})}$. Thus, with overwhelming probability ℓ is sampled before the global optimum.

If the current individual x_t is in *SS* and has less than $a_2 + \frac{n^{\epsilon^*/\beta}}{2}$ 1-bits, then the number of 1-bits after the next mutation is $|x_{t+1}|_1 = |x_t|_1 + Y - (k-Y) = x_t - k + 2Y$, where $Y = Hype(n,k,|x_t|_1)$ denotes the number of 0-bits that are flipped to 1. Since the number of 1-bits in the solution is at least $n \cdot \left(\frac{1}{2} + \epsilon\right)$, there exists a mutation size $k \in O(n^{\epsilon/\beta})$ such that the nearest integer to the expected number of 1-bits after the operation is exactly $a_2 - 1$. Since the hypergeometric distribution is symmetric around (and increasing towards) its mean, the probability that its outcome is in $\mu \pm \sigma$ is constant. Since the standard deviation of Y is $O(\sqrt{n})$, some $c \cdot \sqrt{n}$ integers around the mean share the constant probability in a way that the integers closer to the mean have a larger share. Consequently the integer closest to the mean has a probability of $\Omega(n^{-1/2})$. Considering that the probability of picking the correct mutation size is in the order of $\Omega\left(n^{-\epsilon^*/\beta}\right)$ and $\beta \geq 1$, with probability $\Omega(n^{-\frac{1}{2}-\epsilon^*})$ a solution from *BA* is sampled and accepted due to its higher fitness. In order to improve fitness by increasing the leading ones, at least a particular bit position needs to be flipped which happens with probability $O(1/n)$. Thus, the conditional probability of improving to another *SS* solution before reaching *BA* is at most in the order of $O(n^{-1/2+\epsilon^*})$. In order to reach a solution with $2 \cdot n^{\epsilon^*/\beta}$ leading ones, given that the initial *SS* solution has less than $n^{\epsilon^*/\beta}$ ones, this conditional event needs to occur at least $O(n^{\epsilon^*/\beta})$ times which

happens with probability $n^{-\Omega(n^{\epsilon^*/\beta})}$.

$$\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = \ell, \mathcal{E}_s)$$
$$> (1 - n^{-\Omega(n^{\epsilon/\beta})}) \cdot \Pr(\mathcal{E}_0|\alpha_t = 0, x_t \in BA)$$

Next, we will consider the probability $\Pr(\mathcal{E}_0|\alpha_t = 0, x_t \in BA, \mathcal{E}_s)$. For any BA solution, the only search points with higher fitness are either the global optimum or those belonging to either ℓ or LO. However, there is a local gradient towards LO and ℓ while the improvement to the global optimum has a probability $O(n^{-\Omega(n)})$. Thus, with overwhelming probability the process ends up in either ℓ or LO. Similarly, individuals in LO can only improve into the global optimum or to ℓ thus with overwhelming probability reaches LO before the global optimum.

$$\Pr(\mathcal{E}_0|\alpha_t = 0, x_t \in BA)$$
$$> (1 - n^{-\Omega(n)}) \cdot \Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell)$$

Now we can sequentially replace the conditional probabilities with the bounds obtained up to (8) while combining the factors in the form into $(1 - n^{-\Omega(n^{\epsilon/\beta})})$ via the union bound:

$$\Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell)$$
$$> \frac{1}{2} + \frac{1}{2} \cdot (1 - n^{-\Omega(n^{\epsilon/\beta})}) \cdot \Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell)$$
$$\implies$$
$$\Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell) > (1 - n^{-\Omega(n^{\epsilon/\beta})})$$

We can now conclude our proof by showing that after a reinitialisation the algorithm samples ℓ before the global optimum with overwhelming probability. With probability $1 - e^{-\Omega(n)}$ the newly initialised solutions have $\frac{n}{2} \pm n^{2/3}$ 1-bits in their strings due to Chernoff Bounds on binomially distributed variables [15]. Thus, the Hamming distance to the optimal solution and the SS solutions are in the order of $\Omega(n)$ since all those solutions have at least $a_1 \geq n\left(\frac{1}{2} + \epsilon\right)$ 1-bits. In the iterations until the algorithm samples ℓ (which takes $O(n^2)$ in expectation), the probability of sampling the optimum is in the order of $n^{-\Omega(n)}$. The probability of sampling an initial solution with more than $\frac{n}{2} \pm n^{2/3}$ 1-bits has a larger failure probability than reinitialisation after ℓ is sampled, thus determines the lower bound on the expected number of reinitialisation before the optimum is found as $e^{\Omega(n)}$. □

Next, we focus on the mutation from the local optimum $1^{a_1}0^{n-a_1}$ to any solution on the second slope. Thus, we obtain a lower bound on the runtime by finding how many times the algorithm has to restart to successfully sample the first solution on the second slope. While the theorem holds for all β values, it implies that when $d := a_2 - a_1 = \Omega(n^\delta)$ for any constant $\delta > 0$, superconstant β values lead to expected runtimes that are superpolynomially smaller than the expected times for constant β (Theorem 3). The speed of the adaptive variants lies in their ability to use $\beta = 1$ when the algorithm is stuck at the optima, which increases the probabilty of picking the correct mutation size, while at the same time using a very high $\beta > \log n$ on the second slope where flipping many bits brings a high risk of sampling a solution from the basin of attraction of the local optima.

Theorem 6. *The Fast (1+1) AIS$_\beta$ and the Fast (1+1) AIS$_{s\beta}$ with $\beta \geq 1$ and $\tau > 0$ cannot optimise CliffWithBasin$_{\boldsymbol{a}}$ with $d := a_2 - a_1$ in fewer than $\Omega((d/4)^\beta \binom{n}{d} n^3)$ expected fitness function evaluations.*

Proof. We use the same notation and follow the same arguments in the proof of Theorem 3 to establish that with overwhelming probability a newly initialised solution reaches the local optimum before the optimum. Next, we will consider the conditional probability $\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t = BA, \mathcal{E}_s)$, i.e., given an offspring in BA survived while the parent is removed. The probability that the survived individual $x_t \in BA$ has at least $a_1 + i$ leading ones for any $i \in [d]$ is at most $O(n^{-i} \cdot i^{-\log n})$ since the parent of x_t is ℓ with exactly a_1 leading ones. The probability of decreasing the number of 1-bits in a generation is $\Omega(1)$ while the probability of sampling a solution in SS is $O(n^{-d+i} \cdot (d-i)^{-\log n})$. If a solution from SS is sampled, then it is still necessary that another improvement occurs so that the solution is not removed from the population, which has probability $\Omega(1/n)$ for all points in SS. Once an improvement occurs in BA, the age of the offspring is reset to zero and since individuals in BA have higher fitness than those in SS, the only acceptable solution that will increase the Hamming distance to ℓ is the global optimum. Thus,

$$\Pr(\mathcal{E}_0|\alpha_t = \tau, x_t \in BA, \mathcal{E}_s)$$
$$= \left(1 - \frac{1}{n} \cdot \sum_{i=1}^{d} O(n^{-i} \cdot i^{-\log n}) \cdot O(n^{-d+i} \cdot (d-i)^{-\log n}\right)$$
$$\cdot \Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell \cup LO)$$
$$= (1 - O(n^{-d-1} \cdot (d/4)^{-\log n})) \cdot \Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell \cup LO)$$

Now, we can follow the same arguments as in Theorem 3 and replace the conditional probabilities sequentially to obtain the

$$\left(1 - O\left(n^{-1} \cdot \binom{n}{d}^{-1}\right)\right) \cdot \left(1 - O\left(n^{-d-1} \cdot (d/4)^{-\log n}\right)\right)$$

probability for $\Pr(\mathcal{E}_0|\alpha_t = 0, x_t = \ell)$ which is in the order of

$$\left(1 - O\left((d/4)^{-\log n} \binom{n}{d}^{-1} \cdot n^{-1}\right)\right).$$

Since after a reinitialization it takes $\Omega(n^2)$ to reach ℓ, our claim follows. □

Finally we show that by adapting the parameter β during the run, the algorithms can optimise the CliffWithBasin function efficiently. The adaptive algorithms use the minimal β values at the local optimum that minimise the lower bound from Theorem 6 while increasing the β value on the second slope avoids the exponential runtime proven for fixed $\beta = O(1)$ in Theorem 5.

Theorem 7. *The Adaptive Fast (1+1) AIS$_\beta$ and the Adaptive Fast (1+1) AIS$_{s\beta}$ with the interval $\beta \in [1, (\log n)^2]$, $\rho \geq (12 + \epsilon_1) \cdot n \log n$ and ageing parameter $\tau = \Omega(n^{1+\epsilon_2})$ for an arbitrarily small $\epsilon_1, \epsilon_2 > 0$ optimise CliffWithBasin$_{\boldsymbol{a}}$ with $d := a_2 - a_1$ in $O(d\binom{n}{d} n^3 (\log n)^2)$ expected fitness function evaluations.*

Proof. We will first bound the expected time until the local optimum $1^{a_1}0^{n-a_1}$ is sampled for the first time. When a solution

is randomly initialised, the number of 1-bits in its bit-string is distributed binomially with parameters n and $1/2$. With overwhelming probability, the initial solution has $\frac{n}{2} \pm n^{2/3}$ 1-bits in its string due to Chernoff Bounds on binomially distributed variables [15]. Let $G := \{0,1\}^n \setminus (FS \cup SS \cup BA \cup LO \cup NH)$ be the set of solutions where the CliffWithBasin function is evaluated as $|\frac{n}{2} - \text{OneMax}(x)|^{-1}$. For any solution $x \in G$ the probability of improving the function value is at least $1/2$ given that the hypermutation operator flips only a single bit position. Pessimistically assuming that $\beta = 1$, we can bound the expected time until the algorithm samples a solution with better fitness by $O(\log n)$ due to the probability of flipping a single bit and the expected time until a solution in NH is sampled by $O(n^{2/3} \log n)$ expected function evaluations since with overwhelming probability the initial solution has $n/2 \pm O(n^{2/3})$ 1-bits. Using the law of total expectation as in Theorem 1 we can conclude that the expected time to sample a solution in FS, given that the initial solution is in NH is at most $O(n^2)$ in expectation since all individuals in NH have a Hamming neighbour with better fitness and only $O(n)$ different fitness values. We can then extend the same argument to the progress in FS and establish that it takes $O(n^2)$ iterations after a uniformly random initialisation in expectation to sample the local optimum for the first time.

At the local optimum of CliffWithBasin (i.e., $1^{a_1}0^{n-a_1}$), we will pessimistically assume that we do not find the optimum in ρ steps and have $\beta = 1$ when the ageing triggers. Similarly to the arguments in Theorem 2 the probability that the parent is removed from the population while the offspring survives is a constant. We differ from the previous theorem's argument here since at the same iteration that the parent individual is removed from the population, the offspring must precisely be 1^k0^{n-k} for some $k > a_2$ which occurs if at least $a_2 - a_1$ precise bits are flipped by the hypermutation with probability $\Omega\left((d \log n)^{-1} \binom{n}{d}^{-1}\right)$. In the following iteration with probability $1/n \log n$ we find a second improvement and reset our age to zero and set $\beta = (\log n)^2$. Once a solution of the form 1^k0^{n-k} for $k > a_2$ is sampled, the arguments for the constant probability of finding the optimal solution before sampling a solution with less than a_2 1-bits, follows the arguments in Theorem 2 for the Cliff function. Since the time to reach the local optimum after reinitialisation is in the order of $O(n^2)$ and the probability of improving twice after ageing triggers is $\Omega\left((d \log n)^{-1} \binom{n}{d}^{-1}\right) \cdot (n \log n)^{-1}$, our claim follows. □

Similarly to Theorem 4 if we use $\beta_{\min} = 1 + \epsilon$ for some constant $\epsilon > 0$, the runtime changes by a factor of $d^{2\epsilon}/(\log n)^2$.

6 CONCLUSION

In recent years, power-law mutation operators have gained increasing interest in the randomized search heuristics community. The reason is that, compared to the traditional binomially distributed mutation operators, they have been shown to provide exponential speed-ups at escaping from local optima that require large mutations while still exhibiting asymptotically optimal hillclimbing performance. Heavy-tailed mutations may become detrimental, though, when used together with some form of non-elitism which may be useful to overcome local optima by accepting solutions of lower quality. In this paper we have shown how such disadvantages may be overcome by automatically adapting the power-law distribution during the run so to learn to decrease the mutation rate once local optima have been escaped from.

While we have used standard multimodal benchmark functions with significant structures to explain the scenarios where the adaptive algorithms may considerably improve the performance over static binomial and power-law mutation operators and have provided rigorous evidence of their superiority for all problems where the performance of the heavy-tailed operators is known (and wider classes), our results also easily extend to the NP-Hard Number-Partitioning problem since both ageing and hypermutations are effective for identifying arbitrarily good approximations [9]. Future work should further evaluate experimentally and theoretically the performance of the proposed algorithms on classical combinatorial optimization problems and real-world applications. We note that recently a different self-adjusting mechanism has been proposed in the literature with the aim of increasing the standard bit mutation rate of evolutionary algorithms when detecting local optima [35]. While the mechanism was not designed with the aim of enhancing the non-elitist performance of search heuristics as well as the mutation operator in the presence of local optima, a comparison with our own proposed mechanism should be performed in the near future.

REFERENCES

[1] Anne Auger and Nikolaus Hansen. 2011. Theory of Evolution Strategies: a New Perspective. In *Theory of Randomized Search Heuristics: Foundations and Recent Developments*. World Scientific Publishing Co., Inc., 289–325.

[2] Süntje Böttcher, Benjamin Doerr, and Frank Neumann. 2010. Optimal Fixed and Adaptive Mutation Rates for the LeadingOnes Problem. In *Parallel Problem Solving from Nature (PPSN '10)*. Springer, 1–10. https://doi.org/10.1007/978-3-642-15844-5_1

[3] Frank M. Burnet. 1959. *The Clonal Selection Theory of Acquired Immunity*. Cambridge University Press.

[4] Dogan Corus, Duc-Cuong Dang, Anton V Eremeev, and Per Kristian Lehre. 2017. Level-based analysis of genetic algorithms and other search processes. *IEEE Transactions on Evolutionary Computation* 22, 5 (2017), 707–719.

[5] Dogan Corus, Andrei Lissovoi, Pietro S. Oliveto, and Carsten Witt. 2021. On Steady-State Evolutionary Algorithms and Selective Pressure: Why Inverse Rank-Based Allocation of Reproductive Trials Is Best. *ACM Transactions on Evolutionary Learning and Optimization* 1, 1 (2021).

[6] Dogan Corus and Pietro S. Oliveto. 2017. Standard Steady State Genetic Algorithms Can Hillclimb Faster than Mutation-only Evolutionary Algorithms. *IEEE Transactions on Evolutionary Computation* 22, 5 (2017), 720–732.

[7] Dogan Corus and Pietro S. Oliveto. 2020. On the Benefits of Populations on the Exploitation Speed of Standard Steady-State Genetic Algorithms. *Algorithmica* 82 (2020), 3676–3706.

[8] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2018. Fast Artificial Immune Systems. In *Proc. of PPSN 2018*. 67–78.

[9] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2019. Artificial Immune Systems Can Find Arbitrarily Good Approximations for the NP-hard Number Partitioning Problem. *Artificial Intelligence* 247 (2019), 180–196.

[10] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2020. When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *Theoretical Computer Science* 832 (2020), 166–185.

[11] Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2021. Fast Immune System Inspired Hypermutation Operators for Combinatorial Optimisation. *(To appear in) IEEE Transactions on Evolutionary Computation. https://arxiv.org/abs/2009.00990* (2021), 1–1.

[12] Vincenzo Cutello, Giuseppe Nicosia, Mario Pavone, and Jonathan Timmis. 2007. An Immune Algorithm for Protein Structure Prediction on Lattice Models. *IEEE Transactions on Evolutionary Computation* 11 (2007), 101–117.

[13] Duc-Cuong Dang, Anton Eremeev, and Per Kristian Lehre. 2021. Escaping Local Optima with Non-Elitist Evolutionary Algorithms. In *Proc. of AAAI 2021*. 12275–12283.

[14] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2018. Escaping

Local Optima Using Crossover With Emergent Diversity. *IEEE Transactions on Evolutionary Computation* 22, 3 (2018), 484–497.

[15] Benjamin Doerr. 2019. Probabilistic Tools for the Analysis of Randomized Optimization Heuristics. In *Theory of Randomized Search Heuristics in Discrete Search Spaces*. Springer, Chapter 1, 1–87.

[16] Benjamin Doerr and Carola Doerr. 2015. Optimal Parameter Choices Through Self-Adjustment: Applying the 1/5-Th Rule in Discrete Settings. In *Proc. of GECCO 2015*. 1335–1342.

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

[18] Benjamin Doerr, Carola Doerr, and Johannes Lengler. 2019. Self-adjusting mutation rates with provably optimal success rules. In *Proc. of GECCO 2019*. 1479–1487.

[19] Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. 2017. Fast Genetic Algorithms. In *Proc. of GECCO 2017*. 777–784.

[20] Benjamin Doerr, Andrei Lissovoi, Pietro S. Oliveto, and John A. Warwicker. 2018. On the Runtime Analysis of Selection Hyper-Heuristics with Adaptive Learning Periods. In *Proc. of GECCO 2018*. 1015–1022.

[21] Stefan Droste, Thomas Jansen, and Ingo Wegener. 2002. On the Analysis of the $(1 + 1)$ Evolutionary Algorithm. *Theoretical Computer Science* 276, 1-2 (2002), 51–81.

[22] Tobias Friedrich, Andreas Göbel, Francesco Quinzan, and Markus Wagner. 2018. Heavy-tailed mutation operators in single-objective combinatorial optimization. In *Proc. of PPSN XV*. 134–145.

[23] Tobias Friedrich, Francesco Quinzan, and Markus Wagner. 2018. Escaping Large Deceptive Basins of Attraction with Heavy-tailed Mutation Operators. In *Proc. of GECCO 2018*. 293–300.

[24] Jens Jägersküpper and Tobias Storch. 2007. When the plus strategy outperforms the comma strategy and when not. In *Proc. of FOCI 2007*. 25–32.

[25] Johnny Kelsey and Jonathan Timmis. 2003. Immune inspired somatic contiguous hypermutation for function optimisation. In *Proc. of GECCO 2003*. 207–218.

[26] Stefan Kern, Sibylle D. Müller, Nikolaus Hansen, Dirk Büche, Jiri Ocenasek, and Petros Koumoutsakos. 2004. Learning Probability Distributions in Continuous Evolutionary Algorithms – A Comparative Review. *Natural Computing: An International Journal* 3, 1 (2004), 77–112.

[27] Scott Kirkpatrick, Daniel Gelatt, and Mario P. Vecchi. 1983. Optimization by Simulated Annealing. *Science* (1983), 671–680. Issue 4598.

[28] Per Kristian Lehre and Carsten Witt. 2012. Black-box search by unbiased variation. *Algorithmica* 64 (2012), 623–642. Issue 4.

[29] Andrei Lissovoi, Pietro S. Oliveto, and John A. Warwicker. 2019. On the Time Complexity of Algorithm Selection Hyper-Heuristics for Multimodal Optimisation. In *Proc. of AAAI 2019*. 2322–2329.

[30] Andrei Lissovoi, Pietro S. Oliveto, and John A. Warwicker. 2020. How the Duration of the Learning Period Affects the Performance of Random Gradient Selection Hyper-Heuristics. In *Proc. of AAAI 2020*. 2376–2383.

[31] Wil Michiels, Jan Korst, and Emile Aarts. 2007. *Theoretical aspects of local search*. Springer, Berlin, Heidelberg.

[32] Pietro S. Oliveto, Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenová. 2018. How to Escape Local Optima in Black Box Optimisation: when Non-elitism Outperforms Elitism. *Algorithmica* 80 (2018), 1604—-1633.

[33] Pietro S. Oliveto and Dirk Sudholt. 2014. On the runtime analysis of stochastic ageing mechanisms. In *Proc. of GECCO 2014*. 113–120.

[34] Pietro S. Oliveto, Dirk Sudholt, and Carsten Witt. 2020. A tight lower bound on the expected runtime of standard steady state genetic algorithms. In *Proc. of GECCO 2020*. 1323–1331.

[35] Amirhossein Rajabi and Carsten Witt. 2020. Self-adjusting evolutionary algorithms for multimodal optimization. In *Proc. of GECCO 2020*. 1314–1322.

[36] D. Sudholt. 2017. How Crossover Speeds up Building Block Assembly in Genetic Algorithms. *Evol. Comp.* 25, 2 (2017), 237–274.

[37] Carsten Witt. 2005. Worst-case and Average-case Approximations by Simple Randomized Search Heuristics. In *Proc. of STACS 2005*. 44–56.

[38] Carsten Witt. 2006. Runtime Analysis of the $(\mu+1)$ EA on Simple Pseudo-Boolean Functions. *Evolutionary Computation* 14, 1 (2006), 65–86.