On the Runtime Analysis of the Opt-IA Artificial Immune System

Dogan Corus
Department of Computer Science
University of Sheffield
Sheffield S1 4DP, UK
d.corus@sheffield.ac.uk

Pietro S. Oliveto
Department of Computer Science
University of Sheffield
Sheffield S1 4DP, UK
p.oliveto@sheffield.ac.uk

Donya Yazdani
Department of Computer Science
University of Sheffield
Sheffield S1 4DP, UK
dyazdani1@sheffield.ac.uk

ABSTRACT

We present a time complexity analysis of the Opt-IA artificial immune system (AIS). We first highlight the power and limitations of its distinguishing operators (i.e., hypermutations with mutation potential and ageing) by analysing them in isolation. Recent work has shown that ageing combined with local mutations can help escape local optima on a dynamic optimisation benchmark function. We generalise this result by rigorously proving that ageing leads to considerable speed-ups (compared to evolutionary algorithms (EAs)) on the standard Cliff benchmark function both when using local and global mutations. Unless the *stop at first constructive mutation* (FCM) mechanism is applied, we show that hypermutations require exponential expected runtime to optimise any function with a polynomial number of optima. If instead FCM is used, the expected runtime is at most a linear factor larger than the upper bound achieved for any random local search algorithm using the artificial fitness levels method. Nevertheless, we prove that algorithms using hypermutations can be considerably faster than EAs at escaping local optima. An analysis of the complete Opt-IA reveals that it is efficient on the previously considered functions and highlights problems where the use of the full algorithm is crucial.

CCS CONCEPTS

•Theory of computation → Optimization with randomized search heuristics;

KEYWORDS

Artificial immune systems, Runtime analysis, Theory, Hypermutation, Ageing, Mutation potential, Stop at first constructive mutation

ACM Reference format:
Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. 2017. On the Runtime Analysis of the Opt-IA Artificial Immune System. In *Proceedings of the Genetic and Evolutionary Computation Conference 2017, Berlin, Germany, July 15–19, 2017 (GECCO '17),* 8 pages.
DOI: 10.475/123_4

1 INTRODUCTION

Artificial Immune Systems (AIS) are a class of bio-inspired computing techniques that take inspiration from the immune system of vertebrates [7]. Burnet's clonal selection theory [1] has inspired various AIS for function optimisation. The most popular ones are Clonalg [8], the B-Cell Algorithm [18] and Opt-IA [5].

After numerous successful applications of AIS were reported, a growing body of theoretical work has gradually been built to shed light on the working principles of AIS. While initial work derived conditions that allowed to prove whether an AIS converges or not [4], nowadays rigorous time complexity analyses of AIS are available. Initial runtime analyses focused on studying the performance of typical AIS operators in isolation to explain when and why they are effective. Such studies have been extensively performed for the contiguous somatic hypermutation operator employed by the B-Cell algorithm [2, 14], the inversely proportional hypermutation operator of Clonalg [23, 24] and the ageing operator used by Opt-IA [10, 15, 20]. These studies formed a foundational basis which allowed the subsequent analysis of the complete B-Cell algorithm as used in practice for standard NP-hard problems [13, 17].

Compared to the relatively well understood B-Cell algorithm, the theoretical understanding of other AIS for optimisation is particularly limited. In this paper we consider the complete Opt-IA algorithm [3, 5]. The main distinguishing features of Opt-IA to other AIS is their use of an ageing operator and of hypermutations with mutation potentials [1]. In this work we will first analyse the characteristics of these operators respectively in isolation and afterwards consider a simple, but complete, Opt-IA algorithm. The aim is to highlight function characteristics for which Opt-IA and its main components are particularly effective, hence when it may be preferable to standard Evolutionary Algorithms (EAs).

The idea behind the ageing operator is that old individuals should have a lower probability of surviving compared to younger ones. Ageing was originally introduced as a mechanism to maintain diversity. Theoretical analyses have strived to justify this initial motivation because the new random individuals (introduced to replace old individuals) typically have very low fitness and die out quickly. On the other hand, it is well understood that ageing can be used as a substitute for a standard restart strategy if the whole population dies at the same generation [15] and to escape local optima if most of the population dies except for one survivor that, at the same generation, moves out of the local optimum [20]. This effect was shown for a Random Local Search (RLS) algorithm equipped with ageing on the Balance dynamic optimisation benchmark function. An evolutionary algorithm (EA) using standard bit mutation (SBM) [12, 21] and ageing would not be able to escape the local optima due to their very large basin of attraction. In Section 4 we carefully analyse the ability of ageing to escape local optima on the more general Cliff benchmark function and show that using the

[1]The cloning operator is a common feature in AIS and the hypermacromutation operator is essentially the same as the well studied contiguous somatic mutation operator of the B-Cell algorithm.

operator with both RLS and EAs can make a difference between polynomial and exponential runtimes.

Hypermutation operators are inspired by high mutation rates occurring in the immune system. In Opt-IA the *mutation potential* is linear in the problem size and in different algorithmic variants may either be static or increase by a factor that is proportional to the fitness of the solution (i.e., the b-cell) undergoing the mutation or decrease by a factor that is inversely proportional to the fitness. The theoretical understanding of hypermutations with mutation potential is very limited. To the best of our knowledge the only runtime analysis available is [16], where inversely proportional hypermutations were considered, with and without the *stop at first constructive mutation (FCM)* strategy[2]. The analysis revealed that, without FCM, the operator requires exponential runtime to optimise the standard OneMax function, while by using FCM, the algorithm is efficient. In Section 3 we consider a different hypermutation variant using static mutation potentials and argue that it is just as effective if not superior to other variants. We first show that the use of FCM is essential by rigorously proving that a $(1+1)$EA equipped with hypermutations and no FCM requires exponential time to optimise any function with a polynomial number of optima. We then consider the operator with FCM for any objective function that can be analysed using the artificial fitness level (AFL) method [12, 21] and show an upper bound on its runtime that is at most by a linear factor bigger than the upper bound obtained for any RLS algorithm using AFL. Finally, we use the standard Cliff and Jump benchmark functions to show that hypermutations can achieve considerable speed-ups for escaping local optima compared to well studied EAs.

In Section 5 we consider the complete Opt-IA algorithm. The standard Opt-IA uses both hypermutations and hypermacromutation (both with FCM) mainly because preliminary experimental studies for trap functions indicated that this setting led to the best results [3, 5]. Our analysis reveals that it is unnecessary to use both operators for Opt-IA to be efficient on trap functions. To this end, we will consider the simple version using only static hypermutations as in [3] with the simplification that we allow genotypic duplicates in the population to simplify the analysis and enhance the probabilities of ageing to create copies and escape from local optima. After discussing that the algorithm is efficient for the functions considered in the previous sections, we present a function called HiddenPath, where it is necessary to use both ageing and hypermutations in conjunction, hence the use of Opt-IA in its totality is crucial. Due to space restrictions, some straightforward proofs are omitted from this extended abstract.

2 PRELIMINARIES

In this section we present the standard Opt-IA as applied in [5] for the maximisation of $f : \{0,1\}^n \rightarrow \mathbb{R}$. The algorithm is initialised with a population of μ b-cells generated uniformly at random, each with *age = 0*. In each generation, the algorithm creates a new parent population consisting of *dup* copies of each b-cell (i.e., Cloning) which will be the subject of variation.

The variation stage in Opt-IA uses a *hypermutation operator with mutation potential* sometimes followed by *hypermacromutation* [5], sometimes not [3]. If both are applied, they both act on the clone population (i.e., not in sequence) such that they generate μ mutants each. The number of bits M that are flipped by the hypermutation operator is determined by a function called *mutation potential.* Three different potentials have been considered in the literature: *static* where the number of bits that are flipped is linear in the problem size and does not depend on the fitness function[3], *proportional* (i.e., a linear number of bits are always flipped but increasing proportionally with the fitness of the mutated b-cell) and *inversely proportional* (i.e., to the fitness of the mutated b-cell). The latter potential was previously theoretically analysed in [16]. What is unclear from the literature is whether the M bits to be flipped should be distinct or not and, when using FCM, whether a *constructive mutation* is a strictly improving move or whether a solution of equal fitness suffices. In this paper we will consider the static hypermutation operator with pseudo-code given in Algorithm 1. In particular, the M flipped bits will always be distinct and both kinds of constructive mutation will be considered. At the end of the variation stage all created individuals have $age = 0$ if their fitness is higher than that of their parent cell, otherwise they inherit their parent's age. Then the whole population (i.e., parents and offspring) undergoes the ageing process in which the age of each b-cell is increased by one. Additionally, the ageing operator removes old individuals. Three methods have been proposed in the literature: *static ageing*, which deterministically removes all individuals who exceed age τ, *stochastic ageing*, which removes each individual at each generation with probability p_{die}, and the recently introduced *hybrid ageing* [20], where individuals have a probability p_{die} of dying only once they reach an age of τ. In [20] it was shown that the hybrid version allows to escape local optima, hence we employ this version in this paper and give its pseudo-code in Algorithm 2.

The generation ends with a selection phase. If the total number of b-cells that have survived the ageing operator is larger than μ, then a standard $(\mu+\lambda)$ selection scheme is used with the exception that genotype duplicates are not allowed. If the population size is less than μ, then a birth phase fills the population up to size μ by introducing random b-cells of $age = 0$. In this paper we prefer to allow genotype duplicates because, as we will show, this may only help the ageing operator escape local optima more efficiently.

3 STATIC HYPERMUTATION

The aim of this section is to highlight the power and limitations of the static hypermutation operator in isolation. For this purpose we embed the operator into a minimal AIS framework that uses a population of only one b-cell and creates exactly one clone per generation. The resulting $(1+1)\text{IA}^{\text{hyp}}$, depicted in Algorithm 3, is essentially a $(1+1)$EA that applies the static hypermutation operator instead of using SBM. We will first show that, without the use of FCM, hypermutations are an inefficient variation operator for virtually any optimisation function of interest. From there on we will only consider the operator equipped with FCM. Then the $(1+1)\text{IA}^{\text{hyp}}$ has a runtime that is at most a linear factor larger

[2] An analysis of high mutation rates in the context of population-based evolutionary algorithms was performed in [19].

[3] In [3] the mutation potential is declared to be a constant $0 < c \leq 1$. This is obviously a typo: the authors intended the mutation potential to be cn, where $0 < c \leq 1$.

Algorithm 1 Static hypermutation ($P^{(clo)}, c, n$)

```
1: $M = cn$ (Mutation potential)
2: for all $x_i \in P^{(clo)}$ do
3:     if FCM is not used then
4:         create $y_i$ by flipping $M$ distinct bits selected uniformly at random
5:     else
6:         create $y_i$ by flipping at most $M$ distinct bits selected uniformly at random one after another until a constructive mutation happens
7:     If $f(y_i) > f(x_i)$ then $y_i^{age} = 0$, else $y_i^{age} = x_i^{age}$
8:     Add $y_i$ to $P^{(hyp)}$
```

Algorithm 2 Hybrid ageing ($P^{(t)}, P^{(hyp)}, \mu, \tau$)

```
1: for all $x_i \in (P^{(t)}, P^{(hyp)})$ do
2:     $x_i^{age} = x_i^{age} + 1$
3:     if $x_i^{age} > \tau$ then
4:         remove $x_i$ with probability $p_{die} = 1 - 1/\mu$
```

than that obtained for any RLS algorithm using the artificial fitness levels method. Intuitively, this happens because, if the operator flips some bits that were already set correctly at the beginning of its iteration, then it will perform at most cn useless fitness function evaluations before one hypermutation process is concluded. We formalise this result in Theorem 3.3 when FCM only accepts strict improvements, and formally call the algorithm $(1+1)\text{IA}^{\text{hyp}}_{>}$ in this case. For the $(1+1)\text{IA}^{\text{hyp}}_{\geq}$, where FCM also considers points of equal fitness as constructive solutions, we prove in Theorem 3.4 that the algorithm cannot be too slow compared to the standard RLS$_1$ (i.e., flipping one bit per iteration). We show that the presented results are tight for some standard benchmark functions by proving that the $(1+1)\text{IA}^{\text{hyp}}$ has expected runtimes of $\Theta(n^2 \log n)$ for OneMax and $\Theta(n^3)$ for LeadingOnes, respectively, versus the expected $\Theta(n \log n)$ and $\Theta(n^2)$ fitness function evaluations required by RLS$_1$ [21]. Nevertheless, we conclude the section by showing for the standard benchmark functions Jump and Cliff that the $(1+1)\text{IA}^{\text{hyp}}$ can be particularly efficient on functions with local optima that are generally difficult to escape from.

We start by highlighting the limitations of static hypermutation when FCM is not used. Since $M = cn$ distinct bits have to be flipped at once, the outcome of the hypermutation operator is characterised by a uniform distribution over the set of all solutions which have Hamming distance M to the parent. Since M is linear in n, the size of this set of points is exponentially large and thus the probability of a particular outcome is exponentially small. In the following theorem, we formalise this limitation.

Theorem 3.1. *For any function with a polynomial number of optima, the $(1+1)IA^{hyp}$ without FCM needs expected exponential time to find any of the optima.*

Proof. We analyse the expected time of the last step before an optimal solution is found. To do this we optimistically assume that all the optima are at Hamming distance $M = cn$ from the current

Algorithm 3 $(1+1)IA^{hyp}(c, n)$

```
1: Initialisation ($\mu = 1$)
2: while an optimum is not found do
3:     Cloning: $P^{(clo)}$= Cloning ($P^{(t)}, dup = 1$)
4:     Variation: $P^{(hyp)}$= Static hypermutation ($P^{(clo)}, c, n$)
5:     Selection: If $f(P^{(hyp)}) \geq f(P^{(t)})$, then $P^{(t+1)} := P^{(hyp)}$. Else, $P^{(t+1)} := P^{(t)}$
6:     $t := t + 1$
```

b-cell. Otherwise the probability of reaching an optimum would be zero. Then, given that the number of different points at Hamming distance cn from any point is $\binom{n}{cn}$ and they all are reached with equal probability, the probability of finding this optimum in the last step, for any $c \neq 1$, is $p \leq \frac{poly(n)}{\binom{n}{cn}} \leq \frac{poly(n)}{e^{\Omega(n)}} = e^{-\Omega(n)}$. By a simple waiting time argument, the expected time to reach an optimum is at least $e^{\Omega(n)}$. If $c = 1$, all the bits will be flipped by the operator. As a result the algorithm will only ever see the initial point and its exact reverse. Hence, an optimum is only found if the b-cell is randomly initialised on one of the optima or on the reverse of one of the optima. This happens with probability $p = (2 \cdot poly(n)) / (2^{-n}) = 2^{-\Omega(n)}$ which means the expected time in this case is also at least in the order of $2^{\Omega(n)}$.

□

The theorem explains why poor results were achieved in previous experimental work both on benchmark functions and real world applications such as the hard protein folding problem [3, 5]. The authors indeed state that *"With this policy, however, and for the problems which are faced in this paper, the implemented IA did not provide good results"* [5]. Theorem 3.1 shows that this is the case for any optimisation function of practical interest. In [16] it had already been shown that inversely proportional hypermutations cannot optimise OneMax in polynomial time. Although static hypermutations are the focus of this paper, we point out that Theorem 3.1 can easily be extended to both the inversely proportional hypermutations considered in [16] and to the proportional hypermutations from the literature [3]. From now on we will only consider hypermutations coupled with FCM. We start by showing that hypermutations cannot be too slow compared to local search operators. We first state and prove the following helper lemma.

Lemma 3.2. *The probability that the static hypermutation applied to $x \in \{0,1\}^n$ either evaluates a specific $y \in \{0,1\}^n$ with Hamming distance $k \leq cn$ to x (i.e., event E_y), or that it stops earlier on a constructive mutation (i.e., event E_c) is lower bounded by $Pr\{E_y \vee E_c\} \geq \binom{n}{k}^{-1}$. Moreover, if there are no constructive mutations with Hamming distance smaller than k, then $Pr\{E_y\} = \binom{n}{k}^{-1}$.*

Proof. Since the bits to be flipped are picked without replacement, each successive bit-flip increases the Hamming distance between the current solution and the original solution by one. Without loss of generality, let $\pi = (\pi_1, \pi_2, \ldots, \pi_{cn})$ be the sequence of bit-flips that will be executed by the static hypermutation, which are sampled uniformly from the space of all permutations of n bit positions. The lower bound is based on the fact that the first k bit positions in π have $\binom{n}{k}$ different and equally probable outcomes. Since the only event that can prevent the static hypermutation to evaluate the k-th solution in the sequence is the discovery of a constructive mutation in one of the first $k-1$ evaluations, the probability $Pr\{E_y \vee E_c\}$ is larger than $\binom{n}{k}^{-1}$. If no such constructive mutation exists (i.e. $Pr\{E_c\} = 0$) then, the probability $Pr\{E_y\}$ is exactly equal to $\binom{n}{k}^{-1}$. □

We are ready to show that static hypermutations cannot be too slow compared to any upper bound obtained by applying the artificial fitness levels (AFL) method on the expected runtime of the $(1+1)\text{RLS}_k$ which flips exactly k bits to produce a new solution and applies non-strict elitist selection. AFL require a partition of the search space $\mathcal{X}$ into mutually exclusive sets $\bigcup_{i\in\{1,\ldots,m\}} A_i = \mathcal{X}$ such that $\forall i < j, \quad x \in A_i \wedge y \in A_j \implies f(x) < f(y)$. The expected runtime of a $(1+1)$ algorithm $\mathcal{A}$ with variation operator Var(x): $\mathcal{X} \to \mathcal{X}$ to solve any function defined on $\mathcal{X}$ can be upper bounded via AFL by $E(T) \le \sum_{j=1}^{m} \frac{1}{p_i}$, where $p_i = \min_{x\in A_i}\left(Pr\{\text{Var(x)} \in \bigcup_{j=i+1}^{m} A_j\}\right)$ [12, 21].

Theorem 3.3. *Let $E\left(T^{AFL}_{(1+1)RLS_k}\right)$ be any upper bound on the expected runtime of the $(1+1)RLS_k$ established via the artificial fitness levels method. Then, $E\left(T_{(1+1)IA^{hyp}_{>}}\right) \le cn \cdot E\left(T^{AFL}_{(1+1)RLS_k}\right)$.*

Proof. Let the function $c_j(x)$ for solution $x \in A_i$ return the number of solutions which are at Hamming distance j away from x and belong to set A_k for some $k > i$. The upper bound on the expected runtime of the $(1+1)\text{RLS}_k$ to solve any function obtained by applying the AFL method is: $E\left(T^{AFL}_{(1+1)RLS_k)}\right) \le \sum_{j=1}^{m} \frac{1}{p_i}$, where $p_i = \min_{x\in A_i}\left(c_k(x)/\binom{n}{k}\right)$. Since FCM wastes at most cn bit mutations when it fails to improve, to prove our claim it is sufficient to show that for any current solution $x \in A_i$, the probability that $(1+1)\text{IA}^{\text{hyp}}_{>}$ finds an improvement is at least $c_k(x)/\binom{n}{k}$. The theorem follows from Lemma 3.2 and the definition of a constructive mutation for the $(1+1)\text{IA}^{\text{hyp}}_{>}$, since for each one of the $c_k(x)$ search points, the probability of either finding it or finding a constructive mutation is lower bounded by $\binom{n}{k}^{-1}$. □

A similar result with the additional restriction that $k = 1$ can be obtained for the $(1+1)\text{IA}^{\text{hyp}}_{\ge}$.

Theorem 3.4. $E\left(T_{(1+1)IA^{hyp}_{\ge}}\right) \le cn \cdot E\left(T^{AFL}_{(1+1)RLS_1}\right)$, *where $E\left(T^{AFL}_{(1+1)RLS_1}\right)$ is any upper bound on the expected runtime of the $(1+1)RLS_1$ established via the artificial fitness levels method.*

Proof. The probability that static hypermutation produces a particular Hamming neighbour of the input solution in the first mutation step is $1/n$, which is equal to the probability that RLS_1 produces the same solution. The theorem follows from the fact that in every failure to obtain an improvement in the first step, static hypermutation wastes at most cn fitness evaluations. □

In the following we show that the upper bounds of the previous theorems are tight for well-known benchmark functions.

Theorem 3.5. *The expected runtime of the $(1+1)IA^{hyp}_{>}$ and of the $(1+1)IA^{hyp}_{\ge}$ to optimise OneMax(x):= $\sum_{i=1}^{n} x_i$ is $\Theta(n^2 \log n)$.*

Proof. The upper bounds for the $>$ and $\ge$ FCM selection versions follow respectively from theorems 3.3 and 3.4 since it is easy to derive an upper bound of $O(n \log n)$ for RLS using AFL [21]. For the lower bound of the $>$ FCM selection version, we follow the analysis in Theorem 3 of [16] for inversely proportional hypermutation (IPH) with FCM to solve ZeroMin. The proof there relies on IPH wasting cn function evaluations every time it fails to find an improvement. This is obviously also true for static hypermutations (albeit for a different constant c), hence the proof also applies to our algorithm. The lower bound also holds for the $\ge$ FCM selection algorithm as this algorithm cannot be faster on OneMax by accepting solutions of equal fitness (i.e., the probability of finding an improved solution does not increase). □

We now turn to the LeadingOnes function, which simply returns the number of consecutive 1-bits before the first 0-bit.

Theorem 3.6. *The expected runtime of the $(1+1)IA^{hyp}_{\ge}$ on LeadingOnes:= $\sum_{i=1}^{n}\prod_{j=1}^{i} x_i$ is $\Theta(n^3)$.*

Proof. The upper bound is implied by Theorem 3.4 because AFL gives an $O(n^2)$ runtime of RLS for LeadingOnes [21]. Let $E[f_i]$ be the expected number of fitness evaluations until an improvement is made, considering that the initial solution has i LeadingOnes. The initial solutions consists of i leading 1-bits, followed by a 0-bit and $n-i-1$ bits which each can be either one or zero with equal probability. Let events E_1, E_2 and E_3 be that the first mutation step flips one of the leading ones (with probability i/n), the first 0-bit (with probability $1/n$) or any of the remaining bits (with probability $(n-i-1)/n$), respectively. If E_1 occurs, then the following mutation steps cannot reach any solution with fitness value i or higher and all cn mutation steps are executed. Since no improvements have been achieved, the remaining expected number of evaluations will be the same as the initial expectation (i.e., $E[f_i|E_1] = cn + E[f_i]$). If E_2 occurs, then a new solution with higher fitness value is acquired and the mutation process stops (i.e., $E[f_i|E_2] = 1$). However if E_3 occurs, since the number of leading 1-bits in the new solution is i, the hypermutation operator stops without any improvement (i.e., $E[f_i|E_3] = 1 + E[f_i]$). According to the law of total expectation: $E[f_i] = \frac{i}{n}(cn + E[f_i]) + \frac{1}{n}\cdot 1 + \frac{n-i-1}{n}(1 + E[f_i])$. When, this equation is solved for $E[f_i]$, we obtain, $E[f_i] = icn + n - i$. Since the expected number of consecutive 1-bits that follow the leftmost 0-bit is less than two, the probability of not skipping a level i is $\Omega(1)$. We obtain a lower bound, $\sum_{i=1}^{n} p_i E[f_i] = \Omega(1)\sum_{i=1}^{n}(icn + n - i) = \Omega(n^3)$ by summing over all fitness levels. □

We now focus on establishing when hypermutations may produce considerable speed-ups. The $\text{Jump}_{(k,n)}$ function, introduced in [9], consists of a OneMax slope with a gap of length k bits that needs to be overcome for the optimum to be found. The function is formally defined as :

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

for $n > 1$ and $k \in \{1...n\}$. Mutation-based EAs require $\Theta(n^k)$ function evaluations to optimise the function and recently a linear factor faster upper bound has been proved for standard crossover-based EAs [6]. Hence, EAs require increasing runtimes as the length of the gap increases, from superpolynomial to exponential as soon as $k = \omega(1)$. The following theorem shows that hypermutations allow speed-ups by an exponential factor of $(e/k)^k$, when the jump is hard to perform.

THEOREM 3.7. *The expected runtime of the* $(1+1)IA^{hyp}$ *to solve* JUMP *is at most* $O(\frac{n^{k+1}\cdot e^k}{k^k})$ *with* $cn > k$.

PROOF. The $(1+1)\text{IA}^{\text{hyp}}$ reaches the fitness level $n-k$ (i.e., the local optimum) in $O(n^2 \log n)$ according to Theorem 3.5. All local optima have Hamming distance k to the optimum and the probability that static hypermutation finds the optimum is lower bounded in Lemma 3.2 by $\binom{n}{k}^{-1}$. Hence, the total expected time to find the optimum is, $E(T_{total}) \leq O(n^2 log\, n) + cn\cdot\binom{n}{k} = O\left(\frac{n^{k+1}\cdot e^k}{k^k}\right)$. □

Obviously, hypermutations can jump over large fitness valleys also on functions with other characteristics. For instance the CLIFF_d function was originally introduced to show when non-elitist EAs may outperform elitist ones [11].

$$\text{CLIFF}_d(x) = \begin{cases} \text{ONEMAX}(x) & \text{if ONEMAX}(x) \leq n-d \\ \text{ONEMAX}(x) - d + 1/2 & \text{otherwise} \end{cases}$$

Similarly to JUMP, the function has a ONEMAX slope with a gap of length d bits that needs to be overcome for the optimum to be found. Differently to JUMP though, the local optimum is followed by another ONEMAX slope leading to the optimum. Hence, algorithms that accept a move jumping to the bottom of the cliff, can then optimise the following slope and reach the optimum. While elitist mutation-based EAs obviously have a runtime of $\Theta(n^d)$ (i.e., they do not accept the jump to the bottom of the cliff), the following corollary shows how hypermutations lead to speed-ups that increase exponentially with the distance d between the cliff and the optimum.

COROLLARY 3.8. *The expected runtime of the* $(1+1)IA^{hyp}$ *using FCM for solving* CLIFF *for any* d *is* $O\left(\frac{n^{d+1}\cdot e^d}{d^d}\right)$.

In the next section we will prove that the ageing operator can lead to surprising speed-ups for CLIFF and functions with similar characteristics.

4 AGEING

It is well-understood that the ageing operator can allow algorithms to escape from local optima. This effect was shown on the BALANCE function from dynamic optimisation for an RLS algorithm embedding a hybrid ageing operator [20]. However, for that specific function, an SBM operator would fail to escape, due to the large basin of attraction of the local optima. In this section we highlight the capabilities of ageing in a more general setting (i.e., the standard CLIFF benchmark function) and show that ageing may also be efficient when coupled with SBM.

Ageing allows to escape from a local optimum if one not locally optimal b-cell is created and survives while all the other b-cells die. For this to happen it is necessary that all the b-cells are old and have similar age. This is achieved on a local optimum by creating copies of the locally optimal b-cell (i.e., the b-cells will inherit the age of their parent). Hence, the ability of a mutation operator to create copies enhances this particular capability of the ageing operator. To this end we first consider a modified RLS algorithm that with some constant probability $p = \Omega(1)$ does not flip any bit and implements the ageing operator presented in Algorithm 2. We call this algorithm $(\mu+1)\text{RLS}^{\text{ageing}}$. Apart from making ageing more effective, this slight modification to the standard RLS considerably simplifies the proofs of the statements we wish to make.

The CLIFF benchmark function is generally used to highlight circumstances when non-elitist EAs outperform elitist ones. Algorithms that accept inferior solutions can be efficient for the function by jumping at the bottom of the Cliff and then optimising the ONEMAX slope. This effect was originally shown for the $(1,\lambda)$EA that can optimise the function in approximately $O(n^{25})$ fitness function evaluations if the population size λ is not too large nor too small [11]. This makes the difference between polynomial and exponential expected runtimes compared to elitist EAs (i.e., $\Theta(n^d)$) if the cliff is located far away from the optimum. A smaller, but still exponential, speed-up was recently shown for the population genetics inspired SSWM algorithm with runtime at most $n^d/e^{\Omega(d)}$ [22]. In the previous section we have already shown that hypermutations are faster than SSWM. The following theorem proves a surprising result for the considered $(\mu+1)\text{RLS}^{\text{ageing}}$ for CLIFF. Not only is the algorithm very fast, but our upper bound becomes lowest (i.e., $O(n \log n)$) when the function is most difficult (i.e., when the cliff is located at distance $d = \Theta(n)$ from the optimum).

THEOREM 4.1. *For* $\mu = O(\log n)$, $p < 1$ *a constant and* $\tau = \Theta(n \log n)$, *the* $(\mu+1)RLS^{ageing}$ *solves* CLIFF *in* $O\left(\frac{\mu^2 n^3 \log n}{d^2}\right)$ *if* $d < n/4 - \epsilon n$ *for any constant* $0 < \epsilon < 1/4$.

PROOF. We will follow the proof of Theorem 10 in [20] and adapt the arguments therein to the ONEMAX landscape and to the RLS operator we use. Given that there are $i < \mu$ individuals with j 1-bits in the population, the probability of creating a new individual with j 1-bits is at least $(i/\mu)p$ because the RLS operator creates a copy with a constant probability p. Hence we follow the proof in [20] to show that in $O(\mu n + n \log n)$ expected steps the population climbs up the ONEMAX slope (i.e., samples a solution with $n-d$ 1-bits) and subsequently the whole population will be taken over by the local optimum in $O(\mu \log \mu)$ expected generations. Now we can apply Lemma 5 in [20] to show that in expected $O(\mu^3)$ steps the whole population will have the same age. As a result after another, at most, $\tau = \Theta(n \log n)$ generations the whole population will reach age τ simultaneously because no improvements may occur unless the optimum is found. Overall, the total expected time until the population consists only of local optima with age τ is at most $O(\mu n + n \log n + \mu \log \mu + \mu^3 + \tau) = O(\mu n + n \log n)$. Now we calculate the probability that in the next step one individual jumps to the bottom of the cliff and the rest die in the same generation. The first event happens with probability $(1-p)(d/n) = \Omega\left(\frac{d}{n}\right)$

(i.e., an offspring solution with $n-d+1$ 1-bits is created by flipping one of the d 0-bits in the parent solution). The probability that the rest of the population dies is $1/\mu \cdot (1-1/\mu)^{\mu-1}$. We now require that the survivor creates an offspring with higher fitness (i.e. with $age = 0$) by flipping one of its 0-bits (i.e. it climbs one step up the second slope). This event happens with probability at least $(1-p)(d-1)/(\mu n) = \Omega\left(\frac{d}{\mu n}\right)$ and in the same generation with probability $(1-1/\mu) = \Omega(1)$ the parent of age $\tau+1$ dies due to ageing. Finally, the new solution (i.e. the "safe" individual) takes over the population in $O(\mu \log \mu)$ expected generations by standard arguments. Using Markov's inequality we show that the probability of the take-over happening before any of the new random individuals are improved ϵn times is at least $1-O(\mu\log\mu/n)$. Hence, the overall probability of this series of consecutive events is $\Omega\left(\frac{d}{n}\right) \cdot \frac{1}{\mu}\left(1-\frac{1}{\mu}\right)^{\mu-1} \cdot \Omega\left(\frac{d}{\mu n}\right) \cdot (1-\frac{1}{\mu}) \cdot \left(1-O\left(\frac{\mu\log\mu}{n}\right)\right) = \Omega\left(\frac{d^2}{n^2\mu^2}\right)$ and the expected number of trials (i.e. climbs up and restarts) until we get a survivor which is safe at the bottom of the cliff is $O\left(\frac{n^2\mu^2}{d^2}\right)$. Every time the set of events fails to happen, we wait for another $O(\mu n + n\log n)$ fitness evaluations until the population reaches a configuration where all individuals are locally optimal and have age τ. Once a safe individual has taken over the population, the expected time to find the global optimum will be at most $O(\mu n + n \log n)$. Overall, the total expected time to optimise Cliff is $E(T_{total}) \leq (O(\mu n + n\log n)) \cdot O\left(\frac{n^2\mu^2}{d^2}\right) + O\left(\mu n + n\log n\right) = O\left(\frac{\mu^2 n^3 \log n}{d^2}\right)$. Finally, we consider the probability that the best individual in the population never dies when climbing up the slopes due to ageing. After any higher fitness level is discovered it takes $O(\mu\log\mu)$ generations in expectation and at most $n^{1/2}$ generations with overwhelming probability until the whole population takes over the level. For the first $n-\log n$ levels, the probability of improving a solution is at least $\Omega(\log n/n)$ and the probability that this improvement does not happen in $\tau - n^{1/2} = \Omega(n\log n)$ generations is at most $(1-\Omega(\log n/n))^{\Omega(n\log n)} = e^{-\Omega(\log^2 n)} = n^{-\Omega(\log n)}$. For the remaining fitness levels, the probability of reaching age τ before improving is similarly $(1-\Omega(1/n))^{\Omega(n\log n)} = e^{-\Omega(\log n)} = n^{-\Omega(1)}$. By the union bound over all levels the probability that the best solution is never lost due to ageing is at least $1-o(1)$. We pessimistically assume that the whole optimisation process has to restart if the best individual reaches age τ. However, since at most $1/(1-o(1)) = O(1)$ restarts are necessary in expectation, our bound on the expected runtime holds. □

We now consider the $(\mu+1)\text{EA}^{\text{ageing}}$ which differs from the $(\mu+1)\text{RLS}^{\text{ageing}}$ by using standard bit mutation with mutation rate $1/n$ instead of flipping exactly one bit. SBM allows copying individuals but, since it is a global operator, it can jump back to the local optima from anywhere in the search space with non-zero probability. Nevertheless, the following theorem shows that the algorithm is still very efficient when the cliff is at linear distance from the optimum. Its proof is omitted since it follows similar arguments to those of Theorem 4.1. The main difference in the analysis is that it has to be shown that once the solutions have jumped to the bottom of the cliff they have a good probability of reaching the optimum before jumping back to the top of the cliff.

Algorithm 4 Opt-IA(c, n, μ, τ, dup)

```
1: **Initialisation:** $t = 0$
   Create $P^{(t)} = \{x_1, ..., x_\mu\}$, a population of $\mu$ b-cells uniformly at random
   $x_i^{age} = 0$ for $i = \{1, ...\mu\}$
2: **while** the optimum is not found **do**
3:     $P^{(clo)}$= Cloning ($P^{(t)}, dup$)
4:     $P^{(hyp)}$= Static hypermutation ($P^{clo}, c, n$)
5:     Hybrid ageing ($P^{(t)}, P^{(hyp)}, \mu, \tau$)
6:     Selection: $P^{(t+1)} := (P^{(t)}, P^{(hyp)})$
       a) While $|P^{(t+1)}| > \mu$, remove the b-cell with the lowest fitness breaking ties uniformly at random.
       b) While $|P^{(t+1)}| < \mu$, create a new b-cell uniformly at random.
7:     $t := t + 1$
```

Theorem 4.2. *The $(\mu+1)EA^{ageing}$ with ageing solves Cliff in expected $O(n^{1+\epsilon})$ time if $d = (1/4)(1-c)n$ for some constant $0 < c < 1$, $\tau = \Theta(n\log n)$ and $\mu = \Theta(1)$, where ϵ is an arbitrary constant.*

5 OPT-IA

After analysing the operators separately, we now consider the complete Opt-IA. According to the obtained results and reasonings in the previous sections, the considered Opt-IA, shown in Algorithm 4, uses static hypermutation coupled with FCM as variation operator, hybrid ageing and standard $(\mu+\lambda)$ selection. Also, a mutation is considered constructive if it results in creating an equally fit solution or a better one. The following theorem proves that Opt-IA can efficiently optimise any benchmark function considered previously in this paper. The theorem uses that the ageing parameter τ is set large enough such that no individuals die with high probability before the optima are found.

Theorem 5.1. *The upper bounds on the expected runtime of Opt-IA with ageing parameter τ properly set for the previously considered functions are as follows: $E[T_{OneMax}] \leq O\left(\mu \cdot dup \cdot n^2 \log n\right)$, $E[T_{LO}] \leq O\left(\mu \cdot dup \cdot n^3\right)$, $E[T_{Jump_k}] \leq O\left(\mu \cdot dup \cdot \frac{n^k \cdot e^k}{k^k}\right)$ and $E[T_{Cliff_k}] \leq O\left(\mu \cdot dup \cdot \frac{n^k \cdot e^k}{k^k}\right)$*

Proof. The claims use that if τ is large enough (i.e., $\Omega(n^2)$ for OneMax, LeadingOnes and $\Omega(n^{k+1})$ for Jump$_k$, Cliff$_k$), then with probability $1-o(1)$ the current best solution will never reach age τ and die due to ageing before the optimum is found. For the Opt-IA to lose the current best solution due to ageing, it is necessary that the best solution is not improved for τ generations consecutively. If the improvement probability for any non-optimal solution is at least p_{min} and if the age τ is set to be $p_{min}^{-1} n$, then the probability that a solution will reach age τ before creating an offspring with higher fitness is at most $(1-p_{min})^\tau \leq e^{-n}$. By the union bound it is also exponentially unlikely that this occurs in a polynomial number of fitness levels that need to be traversed before reaching the optimum. Since the suggested τ for each function is larger than the corresponding $p_{min}^{-1} n$, the upper bounds of the $(1+1)\text{IA}^{\text{hyp}}$ (which does not implement ageing) for OneMax, LeadingOnes, Jump and Cliff are valid for Opt-IA when multiplied by $\mu \cdot dup$ to take into account the population and clones. □

5.1 Opt-IA Can Be More Efficient

In this section, we present the function HiddenPath to illustrate a problem where the use of static hypermutation and ageing together is crucial. When either of these two characteristic operators of Opt-IA is not used, the expected runtime is at least superpolynomial. HiddenPath : $\{0,1\}^n \rightarrow \mathbb{R}$ can be described by a series of modifications to the well-know ZeroMax function. The distinguishing solutions are those with at most four 0-bits and those with at least $n-1$ 0-bits, along with $\log n - 2$ solutions in the form $0^k 1^{n-k}$ for $4 \leq k \leq \log n + 1$. The solutions with exactly $n-1$ 0-bits constitute the local optima of HiddenPath. For any $\epsilon < 1$, the HiddenPath function is formally defined as follows:

$$\text{HiddenPath}(x) = \begin{cases} n-\epsilon & \text{if ZeroMax(x)} = 4 \text{ and } x \neq 0^4 1^{n-4} \\ 0 & \text{if ZeroMax(x)} < 4 \text{ or ZeroMax(x)} = n \\ n-\epsilon+\epsilon k/\log n & \text{if } 4 \leq k \leq \log n + 1 \textit{ and } x = 0^k 1^{n-k} \\ n & \text{if ZeroMax}(x) = n-1 \\ \text{ZeroMax}(x) & \textit{otherwise} \end{cases}$$

Since the all 0-bits string returns fitness value zero, there is a drift towards solutions with $n-1$ 0-bits while the global optimum is the $0^{\log n+1} 1^{n-\log n-1}$ bit string. The solutions with exactly four 0-bits work as a net that stops any static hypermutation that has an input solution with less than four 0-bits. The namesake path to the global optimum consists of $\log n - 2$ Hamming neighbours and the first solution on the path has four 0-bits.

Theorem 5.2. *For $c = 1$, $dup = 1$, $\mu = O(\log n)$ and $\tau = \Omega(n^2 \log n)$, Opt-IA needs expected $O(\mu n^9 \tau)$ fitness evaluations to optimise HiddenPath.*

Proof. For convenience we will call any solution of the form $0^k 1^{n-k}$ for $4 \leq k \leq \log n$ an S_p solution and other solutions with i 0-bits S_i solutions. After $O(n \log n)$ generations in expectation an S_{n-1} solution is found by optimising ZeroMax . Assuming the global optimum is not found first, consider the generation when an S_{n-1} solution is found for the first time. Another S_{n-1} solution is created and accepted by Opt-IA with probability at least $1/n$ since it is sufficient to flip the single 1-bit in the first mutation step and any 0-bit in the second. Thus, S_{n-1} solutions take over the population in expected $O(n\mu)$ generations. Since, apart from the optimum, no other solution has higher fitness than S_{n-1} solutions, the population consists only of S_{n-1} solutions after the take over occurs (no solutions die with high probability). We now bound the expected time until all the population has the same age. Considering that the probability of creating another S_{n-1} solution is $\Theta(1/n)$, the probability of creating two copies in one single generation is $\binom{\mu}{2} O\left(\frac{1}{n^2}\right) = O\left(\frac{\log^2 n}{n^2}\right)$. With constant probability this event does not happen in $o(n^2/\log^2 n)$ generations. Conditional on that at most one additional S_{n-1} solution is created in every generation, we can follow a similar argument as in the proof of Theorem 4.1. Hence, we can show that in expected $O(\mu^3 n)$ iterations after the takeover, the whole population reaches the same age.

When the population of S_{n-1} solutions with the same age reaches age τ, with probability $\mu \cdot 1/\mu \cdot (1-(1/\mu))^{2\mu-1}$ a single new clone survives while the rest of the population dies. With probability $1 - O(1/n)$ the survived clone has hypermutated all n bits (i.e., the survived clone is an S_1 solution). In the following generation, the population consists of an S_1 solution and $\mu - 1$ randomly sampled solutions. With probability 1 the S_1 solution produces an S_4 solution via hypermutation. With overwhelming probability the randomly sampled solutions still have fitness value $n/2 \pm O(\sqrt{n})$, hence the S_1 solution is removed from the population while the S_4 b-cell is kept.

We momentarily ignore the event that the hypermutation reaches an S_{n-1} solution which happens with probability $O(1/n^3)$. The population consists of a single S_4 solution and $\mu - 1$ solutions with at most $n/2 + O(\sqrt{n})$ 0-bits. We will now bound the expected time until S_4 solutions take over the population. Clones of any S_4 solutions are also S_4 solutions with $O(1/n)$ probability after hypermutation. Moreover, if the outcome of hypermutation is neither an S_4, an S_p or an S_{n-1} solution, then it is an S_{n-4} solution since all n bit-flips will have been executed. Since S_{n-4} solutions have higher fitness value than the randomly sampled solutions they stay in the population. In the subsequent generation, if hypermutation does not improve an S_{n-4} solution (which happens with probability $O(1/n)$), it executes n bit-flips to create yet another S_4 solution unless the path is found. This feedback causes the number of S_{n-4} and S_4 solutions to double in each generation with constant probability until they collectively take over the population in $O(\log \mu)$ generations in expectation. Then, with constant probability all the S_{n-4} solutions produce an S_4 solution via hypermutation and consequently the population consists only of S_4 solutions. Since the probability that the static hypermutation creates an S_{n-3} solution from an S_{n-4} solution in $O(\log \mu)$ generations is $o(1)$, the take over occurs in expected $O(\log \mu)$ generations with high probability.

Immediately after the S_4 solutions take over, with probability at least $\binom{n}{8}^{-1} = \Omega(1/n^8)$ the first solution on the path to the optimum is discovered, because any S_4 solution can have at most Hamming distance 8 to the bit string $0^4 1^{n-4}$. After $0^4 1^{n-4}$ is added to the population, the best solution on the path is improved with probability $\Omega(1/n)$ by hypermutation and in $O(n \log n)$ generations the global optimum is found. Since all S_p and S_4 solutions have a Hamming distance smaller than $n-3$ and larger than $n/2$ to any S_{n-1} solution, the probability that a local optimum is found before the global optimum is at most $O(n \log n) \cdot \mu n \binom{n}{3}^{-1} = o(1)$. Hence, the previously excluded event has now been taken into account.

The dominating term in the total expected time stems from the $\Omega(1/n^8)$ probability of finding the first solution on the path. We pessimistically assume that we start over when this event does not occur, which implies that the whole process until that point (which takes at most $O(\tau)$ generations in expectation) should be repeated $O(n^8)$ times in expectation. Multiplying with the maximum possible wasted fitness evaluations per generation (μcn), the upper bound is proven. □

The straightforward proofs of the next two theorems are omitted.

Theorem 5.3. *Opt-IA without ageing (i.e., $\tau = \infty$) with $\mu = O(\log n)$ and $dup = O(1)$ cannot optimise HiddenPath in less than $n^{\Omega(\log n)}$ time in expectation.*

Theorem 5.4. *Opt-IA using standard bit mutation (i.e., without static hypermutation) and ageing cannot optimise function* HiddenPath *in less than exponential time.*

5.2 On Trap Functions

In [3], when Opt-IA was originally introduced, the effectiveness of the algorithm was tested for optimising the following simple trap function.

$$\textsc{Simple Trap} = \begin{cases} \frac{a}{z}(z - \textsc{OneMax}(x)) & \text{if } \textsc{OneMax}(x) \le z \\ \frac{b}{n-z}(\textsc{OneMax}(x) - z) & \text{otherwise} \end{cases}$$

where $z \approx n/4$, $b = n - z - 1$ and $3b/2 \le a \le 2b$ and the optimal solution is the 0^n bit string. The experimental results reported that Opt-IA using hypermutation or hypermacromutations could not optimise the trap function at all, already for problem sizes $n \ge 50$. Static hypermutations and hypermacromutations had low success rates even for $n = 20$. The following theorem with straightforward proof shows that Opt-IA can optimise this simple trap function efficiently.

Theorem 5.5. *Opt-IA needs $O(\mu n^2 \log n)$ expected fitness evaluations to optimise the simple trap function with $\tau = \Omega(\mu n^3)$, $c = 1$ and $dup = 1$.*

Given that Opt-IA was tested in [3] also with the parameters suggested by the theorem (i.e., $c = 1$, $dup = 1$, $\tau = \infty$), we speculate that either FCM was mistakingly not used or the stopping criterion (the total number of allowed fitness evaluations, i.e., 5×10^5) was too small. We point out that, for large enough τ also hypermacromutation will have an expected runtime of $O(\mu n^2 \log n)$ for trap functions [14], with or without FCM. In any case, it is not necessary to apply both hypermutations and hypermacromutation together to efficiently solve trap functions, although the inversely proportional hypermutation operator considered in [16] would fail because it cannot flip n bits when on the local optimum.

6 CONCLUSION

We have presented an analysis of the standard Opt-IA artificial immune system. We first highlighted how both the ageing and hypermutation operators may allow to efficiently escape local optima that are particularly hard for standard evolutionary algorithms. We proved that FCM is essential to the hypermutation operator and suggested considering a mutation *constructive* if the produced fitness is at least as good as the previous one. The reason is that for the sake of increasing exploration capabilities, far away points of equal fitness should be attractive. Concerning ageing, we showed for the first time that the operator can be very efficient when coupled with standard bit mutations and hypermutations. To the best of our knowledge, the operator allows the best known runtime (i.e., $O(n \log n)$) for hard Cliff functions. Afterwards, we presented a class of functions where both the characteristics of ageing and hypermutation are crucial, hence Opt-IA is efficient while standard evolutionary algorithms are inefficient even if coupled with one extra AIS operator (either cloning, ageing, hypermutation or contiguous somatic mutation). As a final remark we point out that all the results presented in Section 5, except for Theorem 5.2 for the HiddenPath function, would also hold for the Opt-IA with genotype diversity as used in [3] since the proofs rely on the ageing operator not triggering at all during the optimisation process (i.e., no genotypic copies of individuals are required).

Acknowledgments: The research leading to these results has received funding from the EPSRC under grant agreement no EP/M004252/1.

REFERENCES

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

[2] Dogan Corus, Jun He, Thomas Jansen, Pietro S. Oliveto, Dirk Sudholt, and Christine Zarges. 2016. On Easiest Functions for Mutation Operators in Bio-Inspired Optimisation. *Algorithmica* (2016). DOI:http://dx.doi.org/10.1007/s00453-016-0201-4

[3] Vincenzo Cutello, Giuseppe Nicosia, and Mario Pavone. 2004. Exploring the Capability of Immune Algorithms: A Characterization of Hypermutation Operators. In *Proc. of ICARIS 2004.* 263–276.

[4] Vincenzo Cutello, Giuseppe Nicosia, Mario Romeo, and Pietro S. Oliveto. 2007. On the Convergence of Immune Algorithms. In *Proc. of FOCI 2007.* 409–415.

[5] Vincenzo Cutello, Mario Pavone, and Jonathan Timmis. 2006. An Immune Algorithm for Protein Structure Prediction on Lattice Models. *IEEE Transactions on Evolutionary Computation* 10 (2006), 844–861.

[6] Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. 2016. Emergence of Diversity and Its Benefits for Crossover in Genetic Algorithms. In *Proc. of PPSN 2016.* 890–900.

[7] Leonardo N. de Castro and Jonathan Timmis. 2002. *Artificial Immune Systems: A New Computational Intelligence Paradigm.* Springer-Verlag, Secaucus, NJ, USA.

[8] Leonardo N. de Castro and Fernando J. Von Zuben. 2002. Learning and Optimization Using the Clonal Selection Principle. *IEEE Transaction on Evolutionary Computation* 6, 3 (2002), 239–251.

[9] 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.

[10] Christian Horoba, Thomas Jansen, and Christine Zarges. 2009. Maximal Age in Randomized Search Heuristics with Aging. In *Proc. of GECCO 2009.* 803–810.

[11] Jens Jägersküpper and Tobias Storch. 2007. When the Plus Strategy Outperforms the Comma Strategy and When Not. In *Proc. of FOCI 2007.* 25–32.

[12] Thomas Jansen. 2013. *Analyzing Evolutionary Algorithms: The Computer Science Perspective.* Springer.

[13] Thomas Jansen, Pietro S. Oliveto, and Christine Zarges. 2011. On the Analysis of the Immune-Inspired B-Cell Algorithm for the Vertex Cover Problem. In *Proc. of ICARIS 2011.* 117–131.

[14] Thomas Jansen and Christine Zarges. 2011. Analyzing Different Variants of Immune Inspired Somatic Contiguous Hypermutations. *Theoretical Computer Science* 412, 6 (2011), 517–533.

[15] Thomas Jansen and Christine Zarges. 2011. On the Role of Age Diversity for Effective Aging Operators. *Evolutionary Intelligence* 4, 2 (2011), 99–125.

[16] Thomas Jansen and Christine Zarges. 2011. Variation in Artificial Immune Systems: Hypermutations with Mutation Potential. In *Proc. of ICARIS 2011.* 132–145.

[17] Thomas Jansen and Christine Zarges. 2012. Computing Longest Common Subsequences with the B-Cell Algorithm. In *Proc. of ICARIS 2012.* 111–124.

[18] Johnny Kelsey and Jonathan Timmis. 2003. Immune Inspired Somatic Contiguous Hypermutation for Function Optimisation. In *Proc. of GECCO 2003.* 207–218.

[19] Pietro S. Oliveto, Per Kristian Lehre, and Frank Neumann. 2009. Theoretical analysis of rank-based mutation-combining exploration and exploitation. In *Proc. of CEC 2009.* 1455–1462.

[20] Pietro S. Oliveto and Dirk Sudholt. 2014. On the Runtime Analysis of Stochastic Ageing Mechanisms. In *Proc. of GECCO 2014.* 113–120.

[21] Pietro S. Oliveto and Xin Yao. 2011. Runtime Analysis of Evolutionary Algorithms for Discrete Optimisation. In *Theory of Randomized Search Heuristics: Foundations and Recent Developments.* World Scientific, Chapter 2, 21–52.

[22] Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenova. 2015. First Steps Towards a Runtime Comparison of Natural and Artificial Evolution. In *Proc. of GECCO 2015.* 1455–1462.

[23] Christine Zarges. 2008. Rigorous Runtime Analysis of Inversely Fitness Proportional Mutation Rates. In *Proc. of PPSN X.* 112–122.

[24] Christine Zarges. 2009. On the Utility of the Population Size for Inversely Fitness Proportional Mutation Rates. In *Proc. of FOGA 2009.* 39–46.