Fast Artificial Immune Systems

Dogan Corus, Pietro S. Oliveto, and Donya Yazdani

University of Sheffield, Rigorous Research, UK.
d.corus@sheffield.ac.uk, p.oliveto@sheffield.ac.uk,
dyazdani1@sheffield.ac.uk

Abstract. Various studies have shown that characteristic Artificial Immune System (AIS) operators such as hypermutations and ageing can be very efficient at escaping local optima of multimodal optimisation problems. However, this efficiency comes at the expense of considerably slower runtimes during the exploitation phase compared to standard evolutionary algorithms. We propose modifications to the traditional 'hypermutations with mutation potential' (HMP) that allow them to be efficient at exploitation as well as maintaining their effective explorative characteristics. Rather than deterministically evaluating fitness after each bitflip of a hypermutation, we sample the fitness function stochastically with a 'parabolic' distribution which allows the 'stop at first constructive mutation' (FCM) variant of HMP to reduce the linear amount of wasted function evaluations when no improvement is found to a constant. By returning the best sampled solution during the hypermutation, rather than the first constructive mutation, we then turn the extremely inefficient HMP operator without FCM, into a very effective operator for the standard Opt-IA AIS using hypermutation, cloning and ageing. We rigorously prove the effectiveness of the two proposed operators by analysing them on all problems where the performance of HPM is rigorously understood in the literature.

Keywords: Artificial immune systems, Runtime analysis

1 Introduction

Several Artificial Immune Systems (AIS) inspired by Burnet's clonal selection principle [1] have been developed to solve optimisation problems. Amongst these, Clonalg [2], B-Cell [3] and Opt-IA [4,5] are the most popular. A common feature of these algorithms is their particularly high mutation rates compared to more traditional evolutionary algorithms (EAs). For instance, the *contiguous somatic hypermutations* (CHM) used by the B-Cell algorithm, choose two random positions in the genotype of a candidate solution and flip all the bits in between[1]. This operation results in a linear number of bits being flipped in an average

[1] A parameter may be used to define the probability that each bit in the region actually flips. However, advantages of CHM over EAs have only been shown when all bits in the region flip.

mutation. The *hypermutations with mutation potential* (HMP) used by Opt-IA tend to flip a linear number of bits unless an improving solution is found first (i.e., if no *stop at first constructive mutation* mechanism (FCM) is used, then the operator fails to optimise efficiently any function with a polynomial number of optima [6]).

Various studies have shown how these high mutation rates allow AIS to escape from local optima for which more traditional randomised search heuristics struggle. Jansen and Zarges proved for a benchmark function called Concatenated Leading Ones Blocks (CLOB) an expected runtime of $O(n^2 \log n)$ using contiguous hypermutations versus the exponential time required by EAs relying on standard bit mutations (SBM) since many bits need to be flipped simultaneously to make progress [7]. Similar effects have also been shown on the NP-Hard longest common subsequence [8] and vertex cover [9] standard combinatorial optimisation problems with practical applications where CHM efficiently escapes local optima where EAs (with and without crossover) are trapped for exponential time.

This efficiency on multimodal problems comes at the expense of being considerably slower in the final exploitation phase of the optimisation process when few bits have to be flipped. For instance CHM requires $\Theta(n^2 \log n)$ expected function evaluations to optimise the easy OneMax and LeadingOnes benchmark functions. Indeed it has recently been shown to require at least $\Omega(n^2)$ function evaluations to optimise any function since its expected runtime for its easiest function is $\Theta(n^2)$ [10]. A disadvantage of CHM is that it is *biased*, in the sense that it behaves differently according to the order in which the information is encoded in the bitstring. In this sense the unbiased HMP used by Opt-IA are easier to apply. Also these hypermutations have been proven to be considerably efficient at escaping local optima such as those of the multimodal Jump, Cliff, and Trap benchmark functions that standard EAs find very difficult [6]. This performance also comes at the expense of being slower in the exploitation phase requiring, for instance, $\Theta(n^2 \log n)$ expected fitness evaluations for OneMax and $\Theta(n^3)$ for LeadingOnes.

In this paper we propose a modification to the HMP operator to allow it to be very efficient in the exploitation phases while maintaining its essential characteristics for escaping from local optima. Rather than evaluating the fitness after each bit flip of a hypermutation as the traditional FCM requires, we propose to evaluate it based on the probability that the mutation will be successful. The probability of hitting a specific point at Hamming distance i from the current point, $\binom{n}{i}^{-1}$, decreases exponentially with the Hamming distance for $i < n/2$ and then it increases again in the same fashion. Based on this observation we evaluate each bit following a 'parabolic' distribution such that the probability of evaluating the i_{th} bit flip decreases as i approaches $n/2$ and then increases again. We rigorously prove that the resulting hypermutation operator, which we call P-hype$_{FCM}$, locates local optima asymptotically as fast as Random Local Search (RLS) for any function where the expected runtime of RLS can be proven with the standard artificial fitness levels method. At the same time the operator is still exponentially faster than EAs for the standard multimodal Jump, Cliff, and Trap benchmark functions.

Hypermutations with mutation potential are usually applied in conjunction with Ageing operators in the standard Opt-IA AIS. The power of ageing at escaping local optima has recently been enhanced by showing how it makes the difference between polynomial and exponential runtimes for the Balance function from dynamic optimisation [11]. For very difficult instances of Cliff, ageing even makes RLS asymptotically as fast as any unbiased mutation based algorithm can be on any function [12] by running in $O(n \ln n)$ expected time [6]. However, the power of ageing at escaping local optima is lost when it is used in combination with hypermutations with mutation potential. In particular, the FCM mechanism does not allow the operator to accept solutions of lower quality, thus cancelling the advantages of ageing. Furthermore, the high mutation rates combined with FCM make the algorithm return to the previous local optimum with very high probability. While the latter problem is naturally solved by our newly proposed P-hype$_{FCM}$ that does not evaluate all bit flips in a hypermutation, the former problem requires a further modification to the HMP. The simple modification that we propose is for the operator, which we call P-hype$_{BM}$, to return the best solution it has found if no constructive mutation is encountered. We rigorously prove that Opt-IA then benefits from both operators for all problems where it was previously analysed in the literature, as desired.

2 Preliminaries

Static hypermutations with mutation potential using FCM (i.e., stop at the first constructive mutation) mutate $M = cn$ distinct bits for constant $0 < c \leq 1$ and evaluate the fitness after each bitflip [6]. If an improvement over the original solution is found before the M_{th} bitflip, then the operator stops and returns the improved solution. This behaviour prevents the hypermutation operator to waste further fitness function evaluations if an improvement has already been found. However, for any realistic objective function the number of iterations where there is an improvement constitutes an asymptotically small fraction of the total runtime. Hence, the fitness function evaluations saved due to the FCM stopping the hypermutation have a very small impact on the global performance of the algorithm. Our proposed modified hypermutation operator, called P-hype, instead only evaluates the fitness after each bitflip with a probability that depends on how many bits have already been flipped in the current hypermutation operation. Since previous theoretical analyses have considered $c = 1$ (i.e., $M = n$) [6], we also use this value throughout this paper. Let p_i be the probability that the solution is evaluated after the i_{th} bit has been flipped. The 'parabolic' probability distribution is defined as follows, where the parameter γ

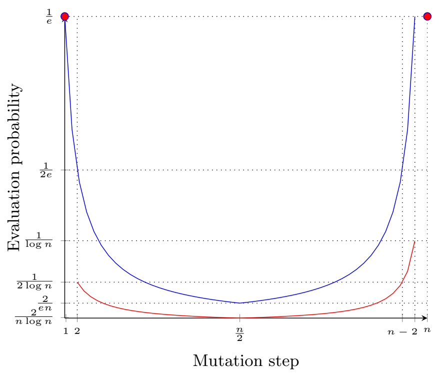

Fig. 1: The parabolic evaluation probabilities (1) for $\gamma = 1/\log n$ and $\gamma = 1/e$.

should be between $0 < \gamma \leq 2$:

$$p_i = \begin{cases} 1/e & \text{for } i = 1 \text{ and } i = n \\ \gamma/i & \text{for } 1 < i \leq n/2 \\ \gamma/(n-i) & \text{for } n/2 < i < n \end{cases} \tag{1}$$

The lower the value of γ, the fewer the expected fitness function evaluations that occur in each hypermutation. For $\gamma = i$ we get the original static hypermutation. On the other hand, with a small enough parameter γ value, the number of wasted evaluations can be dropped to the order of $O(1)$ per iteration instead of the linear amount wasted by the traditional operator when improvements are not found. The resulting hypermutation operator is formally defined as follows.

Definition 1 (P-hype$_{FCM}$). *P-hype$_{FCM}$ flips at most n distinct bits selected uniformly at random. It evaluates the fitness after the i_{th} bitflip with probability p_i (as defined in (1)) and remembers the last evaluation. P-hype$_{FCM}$ stops flipping bits when it finds an improvement; if no improvement is found, it will return the last evaluated solution. If no evaluations are made, the parent will be returned.*

In the next section we will prove its benefits over the standard static HMP with FCM, when incorporated into a (1+1) framework (Algorithm 1). However, in order for the operator to work effectively in conjunction with ageing, a further modification is required. Instead of stopping the hypermutation at the first constructive mutation, we will execute all n mutation steps, evaluate each bitstring with the probabilities in (1) and as the offspring, return the best solution evaluated during the hypermutation or the parent itself if no other bitstrings are evaluated. We will prove that such a modification, which we call P-hype$_{BM}$, may allow the complete Opt-IA to escape local optima more efficiently by P-hype$_{BM}$ producing solutions of lower quality than the local optimum on which the algorithm was stuck while individuals on the local optimum die due to ageing. P-hype$_{BM}$ is formally defined as follows.

Algorithm 1 (1+1) Fast-IA

```
1: Initialise $x$ uniformly at random.
2: while a global optimum is not found do
3:     Create $y = x$, then $y =$ P-hype($y$);
4:     If $f(y) \geq f(x)$, then $x = y$.
5: end while
```

Definition 2 (P-hype$_{BM}$). *P-hype$_{BM}$ flips n distinct bits selected uniformly at random. It evaluates the fitness after the i_{th} bitflip with probability p_i (as defined in (1)) and remembers the best evaluation found so far. P-hype$_{BM}$ returns the mutated solution with the best evaluation found. If no evaluations are made, the parent will be returned.*

For sufficiently small values of the parameter γ only one function evaluation per hypermutation is performed in expectation (although all bits will be flipped). Since it returns the best found one, this solution will be returned by P-hype$_{BM}$ as it is the only one it has encountered. Interestingly, this behaviour is similar to that of the HMP without FCM that also evaluates one point per hypermutation and returns it. However, while HMP without FCM has exponential expected runtime for any function with a polynomial number of optima [6], we will show in the following sections that P-hype$_{BM}$ can be very efficient. From this point of view, P-hype$_{BM}$ is as a very effective way to perform hypermutations with mutation potential without FCM.

In Section 4, we consider P-hype$_{BM}$ in the complete Opt-IA framework [4,5,6] hence analyse its performance combined with cloning and ageing. The algorithm which we call Fast Opt-IA, is depicted in Algorithm 2. We will use the *hybrid ageing* operator as in [6,11], which allows us to escape local optima. Hybrid ageing removes candidate solutions (i.e. b-cells) with probability $p_{die} = 1 - (1/(\mu + 1))$ once they have passed an age threshold τ. After initialising a population of μ b-cells with $age = 0$, at each iteration the algorithm creates dup copies of each b-cell. These copies are mutated by the P-hype operator, creating a population of mutants called P^{hyp} which inherit the age of their parents if they do not improve the fitness; otherwise their age will be set to zero. At the next step, all b-cells with $age \geq \tau$ will be removed from both populations with probability p_{die}. If less than μ individuals have survived ageing, then the population is filled up with new randomly generated individuals. At the selection phase, the best μ b-cells are chosen to form the population for the next generation.

3 Fast Hypermutations

We start our analysis by relating the expected number of fitness function evaluations to the expected number of P-hype operations until the optimum is found. The following result holds for both P-hype operators. The lemma quantifies the number of expected fitness function evaluations which are wasted by a hypermutation operation.

Algorithm 2 Fast Opt-IA

```
1: Initialise a population of $\mu$ b-cells, $P$, created uniformly at random;
2: for each $x \in P$ set $x^{age} = 0$.
3: while a global optimum is not found do
4:     for each $x \in P$ set $x^{age} = x^{age} + 1$;
5:     for  $dup$ times for each $x \in P$  do
6:         $y =$ P-hype($x$);
7:         if $f(y) > f(x)$ then  $y^{age} = 0$ else $y^{age} = x^{age}$;
8:         Add $y$ to $P^{hyp}$.
9:     end for
10:    Add $P^{hyp}$ to $P$, set $P^{hyp} = \emptyset$;
11:    for each $x \in P$ if $x^{age} \geq \tau$ then remove $x$ with probability $p_{die}$;
12:    if $|P| < \mu$ then  add $\mu - |P|$ solutions to $P$ with age zero generated uniformly
       at random;
13:    if $|P| > \mu$ then  remove $|P| - \mu$ solutions with the lowest fitness from $P$ breaking
       ties uniformly at random.
14: end while
```

Lemma 1. *Let T be the random variable denoting the number of P-hype operations applied until the optimum is found. Then, the expected number of total function evaluations is at most: $E[T] \cdot O(1 + \gamma \log n)$.*

Proof. Let the random variable X_i for $i \in [T]$ denote the number of fitness function evaluations during the ith execution of P-hype. Additionally, let the random variable X_i' denote the number of fitness function evaluations at the i_{th} operation assuming that no improvements are found. For all i it holds that $X_i \preceq X_i'$ since finding an improvement can only decrease the number of evaluations. Thus, the total number of function evaluations $E[\sum_{i=1}^{T} X_i]$ can be bounded above by $E[\sum_{i=1}^{T} X_i']$ which is equal to $E[T] \cdot E[X']$ due to Wald's equation [13] since X_i' are identically distributed and independent from T.

We now write the expected number of fitness function evaluations in each operation as the sum of n indicator variables Y_i for $i \in [n]$ denoting whether an evaluation occurs after the ith bit mutation. Referring to the probabilities in (1), we get, $E[X] = E\left[\sum_{i=1}^{n} Y_i\right] = \sum_{i=1}^{n} Pr\{Y_i = 1\} = \frac{1}{e} + \frac{1}{e} + 2\sum_{i=2}^{n/2} \gamma \frac{1}{i} \leq \frac{2}{e} + 2\gamma\left(\ln n/2 - 1\right)$. □

In Lemma 1, γ appears as a multiplicative factor in the expected runtime measured in fitness function evaluations. An intuitive lower bound of $\Omega(1/\log n)$ for γ can be inferred since smaller mutation rates will not decrease the runtime. While a smaller γ does not decrease the asymptotic order of expected evaluations per operation, in Section 4 we will provide an example where a smaller choice of γ reduces $E[T]$ directly. For the rest of our results though, we will rely on $E[T]$ being the same as for the traditional static hypermutations with FCM while the number of wasted fitness function evaluations decreases from n to $O(1+\gamma \log n)$.

Table 1: Expected runtimes of the standard (1+1) EA and (1+1) IA$^{\text{hyp}}$ versus the expected runtime of the (1+1) Fast-IA. For $\gamma = O(1/\log n)$, the (1+1) Fast-IA is asymptotically at least as fast as the (1+1) EA and faster by a linear factor compared to the (1+1) IA$^{\text{hyp}}$ for the unimodal and trap functions. For not too large jump and cliff sizes (i.e. $o(n/\log n)$), the (1+1) Fast-IA has an asymptotic speed up compared to the (1+1) IA$^{\text{hyp}}$ for the same parameter setting. For not too small jump/cliff sizes both AIS are much faster than the (1+1) EA.

Function	(1+1) EA	(1+1) IA$^{\text{hyp}}$	(1+1) Fast-IA
OneMax	$\Theta(n \log n)$ [14]	$\Theta(n^2 \log n)$ [6]	$\Theta\left(n \log n\left(1+\gamma \log n\right)\right)$
LeadingOnes	$\Theta(n^2)$ [14]	$\Theta(n^3)$ [6]	$\Theta\left(n^2\left(1+\gamma \log n\right)\right)$
Trap	$\Theta(n^n)$ [14]	$\Theta(n^2 \log n)$ [6]	$\Theta\left(n \log n\left(1+\gamma \log n\right)\right)$
Jump$_{d>1}$	$\Theta(n^d)$ [14]	$O(n\binom{n}{d})$ [6]	$O\left((d/\gamma) \cdot (1+\gamma \log n) \cdot \binom{n}{d}\right)$
Cliff$_{d>1}$	$\Theta(n^d)$ [15]	$O(n\binom{n}{d})$ [6]	$O\left((d/\gamma) \cdot (1+\gamma \log n) \cdot \binom{n}{d}\right)$

We will now analyse the simplest setting where we can implement P-hype. The (1+1) Fast-IA keeps a single individual in the population and uses P-hype to perturb it at every iteration. The performance of the (1+1) IA$^{\text{hyp}}$, a similar barebones algorithm using the classical static hypermutation operator has recently been related to the performance of the well-studied Randomised Local Search algorithm (RLS) [6]. RLS$_k$ flips exactly k bits of the current solution to sample a new search point, compares it with the current solution and continues with the new one unless it is worse. According to Theorem 3.3 and Theorem 3.4 of [6], any runtime upper bound for RLS obtained via Artificial Fitness Levels (AFL) method also holds for the (1+1) IA$^{\text{hyp}}$ with an additional factor of n (e.g., an upper bound of $O(n)$ for RLS derived via AFL translates into an upper bound of $O(n^2)$ for the (1+1) IA$^{\text{hyp}}$). The following theorem establishes a similar relationship between RLS and the (1+1) Fast-IA with a factor of $O(1+\gamma \log n)$ instead of n. In the context of the following theorem, (1+1) Fast-IA$_{\geq}$ denotes the variant of (1+1) Fast-IA which considers an equally good solution as constructive while (1+1) Fast-IA$_{>}$ stops the hypermutation only if a solution strictly better than the parent is sampled.

Theorem 1. *Let $E\left(T_A^{AFL}\right)$ be any upper bound on the expected runtime of algorithm A established by the artificial fitness levels method. Then* $E\left(T^{AFL}_{\textit{(1+1) Fast-IA}_{>}}\right) \leq E\left(T^{AFL}_{\text{(1+1) }RLS_k}\right) \cdot k/\gamma \cdot O(1+\gamma \log n)$*. Moreover, for the special case of $k=1$, $E\left(T^{AFL}_{\textit{(1+1) Fast-IA}_{\geq}}\right) \leq E\left(T^{AFL}_{\text{(1+1) }RLS_1}\right) \cdot O(1+\gamma \log n)$ also holds.*

Apart from showing the efficiency of the (1+1) Fast-IA, the theorem also allows easy achievements of upper bounds on the runtime of the algorithm, by just analysing the simple RLS. For $\gamma = O(1/\log n)$, Theorem 1 implies the upper bounds of $O(n \log n)$ and $O(n^2)$ for classical benchmark functions OneMax and LeadingOnes respectively (see Table 1). Both of these bounds are asymptotically tight since each function's unary unbiased black-box complexity is in the same order as the presented upper bound [12].

Corollary 1. *The expected runtimes of the (1+1) Fast-IA to optimise* $\textsc{OneMax}(x) := \sum_{i=1}^{n} x_i$ *and* $\textsc{LeadingOnes} := \sum_{i=1}^{n} \prod_{j=1}^{i} x_j$ *are respectively* $O\left(n \log n \left(1 + \gamma \log n\right)\right)$ *and* $O(n^2 \left(1 + \gamma \log n\right))$*. For* $\gamma = O(1/\log n)$ *these bounds reduce to* $\Theta(n \log n)$ *and* $\Theta(n^2)$*.*

P-hype samples the complementary bit-string with probability one if it cannot find any improvements. This behaviour allows an efficient optimisation of the deceptive Trap function which is identical to OneMax except that the optimum is in 0^n. Since n bits have to be flipped to reach the global optimum from the local optimum, evolutionary algorithms based on standard bit mutation require exponential runtime with overwhelming probability [16]. By evaluating the sampled bitstrings stochastically, the (1+1) Fast-IA provides up to a linear speed-up for small enough γ compared to the (1+1) IA^{hyp} on Trap as well.

Theorem 2. *The expected runtime of the (1+1) Fast-IA to optimise* Trap *is* $\Theta(n \log n \left(1 + \gamma \log n\right))$*.*

The results for the (1+1) IA^{hyp} on Jump and Cliff functions [6] can also be adapted to the (1+1) Fast-IA in a straightforward manner, even though they fall out of the scope of Theorem 1. Both $\textsc{Jump}_d$ and $\textsc{Cliff}_d$ have the same output as OneMax for bitstrings with up to $n - d$ 1-bits and the same optimum 1^n. For solutions with the number of 1-bits between $n-d$ and n, Jump has a reversed OneMax slope creating a gradient towards $n - d$ while Cliff has a slope heading toward 1^n even though the fitness values are penalised by an additive factor d. Being designed to accomplish larger mutations, the performance of hypermutations on Jump and Cliff functions is superior to standard bit mutation [6]. This advantage is preserved for the (1+1) Fast-IA as seen in the following theorem.

Theorem 3. *The expected runtime of the (1+1) Fast-IA to optimise* $\textsc{Jump}_d$ *and* $\textsc{Cliff}_d$ *is* $O\left((d/\gamma) \cdot (1 + \gamma \log n) \cdot \binom{n}{d}\right)$*.*

For Jump and Cliff, the superiority of the (1+1) Fast-IA in comparison to the deterministic evaluations scheme depends on the function parameter d. If $\gamma = \Omega(1/\log n)$, the (1+1) Fast-IA performs better for $d = o(n/\log n)$ while the deterministic scheme (i.e., (1+1) IA^{hyp}) is preferable for larger d. However, for small d the difference between the runtimes can be as large as a factor of n in favor of the (1+1) Fast-IA while, even for the largest d, the difference is less than a factor of $\log n$ in favor of the deterministic scheme. Here we should also note that for $d = \omega(n/\log n)$ the expected time is exponentially large for both algorithms (albeit considerably smaller than that of standard EAs) and the $\log n$ factor has no realistic effect on the applicability of the algorithm.

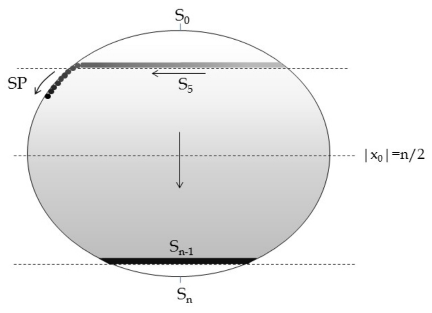

Fig. 2: HiddenPath [6]

4 Fast Opt-IA

In this section we will consider the effect of our proposed evaluation scheme on the complete Opt-IA algorithm. The distinguishing characteristic of the Opt-IA algorithm is its use of the ageing and hypermutation operators. In [6] a fitness function called HiddenPath (Fig. 2) was presented where the use of both operators is necessary to find the optimum in polynomial time. The function HiddenPath provides a gradient to a local optimum, which allows the hypermutation operator to find another gradient which leads to the global optimum but situated on the opposite side of the search space (i.e., nearby the complementary bitstrings of the local optima). However, the ageing operator is necessary for the algorithm to accept worsening; otherwise the second gradient is not accessible. To prove our upper bound, we can follow the same proof strategy in [17], which established an upper bound of $O(\tau\mu n + \mu n^{7/2})$ for the expected runtime of the traditional Opt-IA on HiddenPath. We will see that Opt-IA benefits from an $n/\log n$ speedup due to P-hype.

Theorem 4. *The Fast Opt-IA needs $O(\tau\mu + \mu n^{5/2} \log n)$ fitness function evaluations in expectation to optimise* HiddenPath *with $\mu = O(\log n)$, $dup = 1$, $1/(4 \ln n) \geq \gamma = \Omega(1/\log n)$ and $\tau = \Omega(n \log^2 n)$.*

HiddenPath was artificially constructed to fit the behaviour of the Opt-IA to illustrate its strengths. One of those strengths was the ageing mechanism's ability to escape local optima in two different ways. First, it allows the algorithm to restart with a new random population after it gets stuck at a local optimum. Second, ageing allows individuals with worse fitness than the current best to stay in the population when all the current best individuals are removed by the ageing operator in the same iteration. If an improvement is found soon after the worsening is accepted, then this temporary non-elitist behaviour allows the algorithm to follow other gradients which are accessible by variation from the local optima but leads away from them. On the other hand, even though it is coupled with ageing in the Opt-IA, the FCM mechanism does not allow worsenings.

More precisely, for the hypermutation with FCM, the complementary bit-string of the local optimum is sampled with probability 1 if no other improvements are found. Indeed, HiddenPath was designed to exploit this high probability. However, by only stopping on improving mutations, the traditional hypermutations with FCM do not allow, in general, to take advantage of the power of ageing at escaping local optima. For instance, for the classical benchmark function $\textsc{Cliff}_d$ with parameter $d = \Theta(n)$, hypermutation with FCM turned out to be a worse choice of variation operator to couple with ageing than both local search and standard-bit-mutation [17]. Ageing coupled with RLS and SBM can reach the optimum by local moves, which respectively yields upper bounds of $O(n \log n)$ and $O(n^{1+\epsilon} \log n)$ for arbitrarily small constant ϵ on their runtimes. However, hypermutations with FCM require to increase the number of 1-bits in the current solution by d at least once before the hypermutation stops. This requirement implies the following exponential lower bound on the runtime regardless of the evaluation scheme (as long as the hypermutation only stops on a constructive mutation).

Theorem 5. *Fast Opt-IA using P-hype$_{FCM}$ requires at least $2^{\Omega(n)}$ fitness function evaluations in expectation to find the optimum of $\textsc{Cliff}_d$ for $d = (1-c)n/4$, where c is a constant $1 > c > 0$.*

The following theorem will demonstrate how P-hype$_{BM}$, that, instead of stopping the hypermutation at the first constructive mutation, will execute all n mutation steps, evaluates each bitstring with the probabilities in (1) and return the best found solution, allows ageing and hypermutation to work in harmony in Opt-IA.

Theorem 6. *Fast Opt-IA using P-hype$_{BM}$ with $\mu = 1$, $dup = 1$, $\gamma = 1/(n \log^2 n)$ and $\tau = \Theta(n \log n)$ needs $O(n \log n)$ fitness function evaluations in expectation to optimise $\textsc{Cliff}$ with any linear $d \le n/4 - \epsilon$ for an small constant ϵ.*

Note that the above result requires a γ in the order of $\Theta(1/(n \log^2 n))$, while Lemma 1 implies that any $\gamma = \omega(1/\log n)$ would not decrease the expected number of fitness function evaluations below the asymptotic order of $\Theta(1)$. However, having $\gamma = 1/(n \log^2 n)$ allows Opt-IA with constant probability to complete its local search before any solution with larger Hamming distance is ever evaluated. In Theorem 6, we observe that this opportunity allows the Opt-IA to hillclimb the second slope before jumping back to the local optima. The following theorem rigorously proves that a very small choice for γ in this case is necessary (i.e., $\gamma = \Omega(1/\log n)$ leads to exponential expected runtime).

Theorem 7. *At least $2^{\Omega(n)}$ fitness function evaluations in expectation are executed before the Fast Opt-IA using P-hype$_{BM}$ with $\gamma = \Omega(1/\log n)$ finds the optimum of $\textsc{Cliff}_d$ for $d = (1-c)n/4$, where c is a constant $1 > c > 0$.*

5 Conclusion

Due to recent analyses of increasingly realistic evolutionary algorithms, higher mutation rates, naturally present in artificial immune systems, than previously recommended or used as a rule of thumb, are gaining significant interest in the evolutionary computation community [18,19,20,21].

We have presented two alternative 'hypermutations with mutation potential' operators, P-hype$_{FCM}$ and P-hype$_{BM}$ and have rigorously proved, for several significant benchmark problems from the literature, that they maintain the exploration characteristics of the traditional operators while outperforming them up to linear factor speedups in the exploitation phase.

The main modification that allows to achieve the presented improvements is to sample the solution after the i_{th} bitflip stochastically with probability roughly $p_i = \gamma/i$, rather than deterministically with probability one. The analysis shows that the parameter γ can be set easily. Concerning P-hype$_{FCM}$, that returns the first sampled constructive mutation and is suggested to be used in isolation, any $\gamma = O(1/\log(n))$ allows optimal asymptotical exploitation time (based on the unary unbiased black box complexity of OneMax and LeadingOnes) while maintaining the traditional exploration capabilities. Concerning P-hype$_{BM}$, which does not use FCM and is designed to work harmonically with ageing as in the standard Opt-IA, considerably lower values of the parameter (i.e., $\gamma = 1/(n \log^2 n)$) are required to escape from difficult local optima efficiently (eg. Cliff) such that the hypermutations do not return to the local optima with high probability. While these low values for γ still allow optimal asymptotic exploitation in the unbiased unary black box sense, they considerably reduce the capability of the operator to perform the large jumps required to escape the local optima of functions with characteristics similar to Jump, i.e., where ageing is ineffective due to the second slope of decreasing fitness. Future work may consider an adaptation of the parameter γ to allow it to automatically increase and decrease throughout the run [22,23]. Furthermore, the performance of the proposed operators should be evaluated for classical combinatorial optimisation problems and real-world applications.

References

1. Frank M. Burnet. *The Clonal Selection Theory of Acquired Immunity.* Cambridge University Press, 1959.
2. Leonardo N. de Castro and Fernando J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Trans. Evol. Comp.*, 6(3):239–251, 2002.
3. Johnny Kelsey and Jonathan Timmis. Immune inspired somatic contiguous hypermutation for function optimisation. In *Proc. of GECCO 2003*, pages 207–218, 2003.
4. Vincenzo Cutello, Giuseppe Nicosia, Mario Pavone, and Jonathan Timmis. An immune algorithm for protein structure prediction on lattice models. *IEEE Trans. Evol. Comp.*, 11(1):101–117, 2007.
5. Vincenzo Cutello, Giuseppe Nicosia, and Mario Pavone. A hybrid immune algorithm with information gain for the graph coloring problem. In *Proc. of GECCO 2003*, pages 171–182, 2003.
6. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. On the runtime analysis of the Opt-IA artificial immune system. In *Proc. of GECCO 2017*, pages 83–90, 2017.

7. Thomas Jansen and Christine Zarges. Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theor. Comp. Sci.*, 412(6):517 – 533, 2011.
8. Thomas Jansen and Christine Zarges. Computing longest common subsequences with the B-Cell Algorithm. In *Proc. of ICARIS 2012*, pages 111–124, 2012.
9. Thomas Jansen, Pietro S. Oliveto, and Christine Zarges. On the analysis of the immune-inspired B-Cell algorithm for the vertex cover problem. In *Proc. of ICARIS 2011*, pages 117–131, 2011.
10. Dogan Corus, Jun He, Thomas Jansen, Pietro S. Oliveto, Dirk Sudholt, and Christine Zarges. On easiest functions for mutation operators in bio-inspired optimisation. *Algorithmica*, 78(2):714–740, 2016.
11. Pietro S. Oliveto and Dirk Sudholt. On the runtime analysis of stochastic ageing mechanisms. In *Proc. of GECCO 2014*, pages 113–120, 2014.
12. Per Kristian Lehre and Carsten Witt. Black-box search by unbiased variation. *Algorithmica*, 64(4):623–642, Dec 2012.
13. Michael Mitzenmacher and Eli Upfal. *Probability and Computing: Randomized Algorithms and Probabilistic Analysis.* Cambridge University Press, 2005.
14. Stefan Droste, Thomas Jansen, and Ingo Wegener. On the analysis of the (1+ 1) evolutionary algorithm. *Theor. Comp. Sci.*, 276(1-2):51–81, 2002.
15. Tiago Paixão, Jorge Pérez Heredia, Dirk Sudholt, and Barbora Trubenová. Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713, 2017.
16. Pietro S. Oliveto and Xin Yao. Runtime analysis of evolutionary algorithms for discrete optimization. In Anne Auger and Benjamin Doerr, editors, *Theory of Randomized Search Heuristics*, pages 21–52. World Scientific, 2011.
17. Dogan Corus, Pietro S. Oliveto, and Donya Yazdani. When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *ArXiv e-prints*, 2018. http://arxiv.org/abs/1804.01314.
18. Pietro S. Oliveto, Per Kristian Lehre, and Frank Neumann. Theoretical analysis of rank-based mutation-combining exploration and exploitation. In *Proc. of CEC 2009*, pages 1455–1462, 2009.
19. Benjamin Doerr, Huu Phuoc Le, Régis Makhmara, and Ta Duy Nguyen. Fast genetic algorithms. In *Proc. of GECCO 2017*, pages 777–784, 2017.
20. Dogan Corus and Pietro S. Oliveto. Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Trans. Evol. Comp.*, 2017.
21. Duc-Cuong Dang, Tobias Friedrich, Timo Kötzing, Martin S. Krejca, Per Kristian Lehre, Pietro S. Oliveto, Dirk Sudholt, and Andrew M. Sutton. Emergence of diversity and its benefits for crossover in genetic algorithms. *to appear in IEEE Trans. Evol. Comp.*, 2017.
22. Benjamin Doerr, Andrei Lissovoi, Pietro S. Oliveto, and John Alasdair Warwicker. On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proc. of GECCO 2018*. ACM, 2018, To Appear.
23. Benjamin Doerr and Carola Doerr. Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709, 2018.
24. Robert J Serfling. Probability inequalities for the sum in sampling without replacement. *The Annals of Statistics*, pages 39–48, 1974.
25. Anne Auger and Benjamin Doerr. *Theory of Randomized Search Heuristics: Foundations and Recent Developments.* World Scientific Publishing Co., Inc., 2011.

A Appendix

This appendix contains additional material to be read at the discretion of the reviewers. It is not necessary to understand the main part of the paper (the first 12 pages, which is the paper we submit for a possible publication in the conference proceedings), but it allows to check the correctness of the mathematical results. This appendix thus has a similar role as providing the source code in an experimental publication. If this submission is accepted for PPSN, we will publish a preprint containing all proofs, so that also the readers of the proceedings have access to this material.

The benchmark functions analysed in the paper are formally defined as follows:

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

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

$$\textsc{HiddenPath}(x) = \begin{cases} n - \epsilon + \frac{\sum_{i=n-4}^n (1-x_i)}{n} & \text{if } \sum_{i=1}^n (1-x_i) = 5 \text{ and } x \ne 1^{n-5}0^5 \\ 0 & \text{if } \sum_{i=1}^n (1-x_i) < 5 \\ 0 & \text{if } \sum_{i=1}^n (1-x_i) = n \\ n - \epsilon + \epsilon k / \log n & \text{if } 5 \le k \le \log n + 1 \text{ and } x = 1^{n-k}0^k \\ n & \text{if } \sum_{i=1}^n (1-x_i) = n-1 \\ \sum_{i=1}^n (1-x_i) & \text{otherwise.} \end{cases}$$

We made use of the following theorem by Serfling which provides an upper bound on the outcome of a hypergeometric distribution. Consider a set $C := \{c_1, \ldots, c_n\}$ consisting of n elements, with $c_i \in R$ where c_{min} and c_{max} are the smallest and largest elements in C respectively. Let $\bar{\mu} := (1/n) \sum_{j=1}^n c_i$, be the mean of C. Let $1 \le i \le k \le n$ and X_i denote the ith draw without replacement from C and $\bar{X} := (1/k) \sum_{j=1}^k X_i$ the sample mean.

Theorem 8 (Serfling [24]). *For $1 \le k \le n$, and $\lambda > 0$*

$$Pr\left\{ \sqrt{k}(\bar{X} - \bar{\mu}) \ge \lambda \right\} \le \exp\left(-\frac{2\lambda^2}{(1 - f_k^*)(c_{max} - c_{min})^2} \right)$$

where $f_k^* := \frac{k-1}{n}$.

Artificial Fitness Levels (AFL) is a method to derive upperbound on the expected runtime of a (1+1) algorithm [25]. AFL divides the search space into m mutually exclusive partitions $A_1 \cdots, A_m$ such that all the points in A_i have less fitness than points which belong to A_j for all $j > i$. The last partition, A_m only includes the global optimum. If p_i is the smallest probability that an individual belonging to A_i mutates to an individual belonging to A_j such that $i < j$, then the expected time to find the optimum is $E(T) \leq \sum_{i=1}^{m-1} 1/p_i$. This method is used in the below theorem.

Theorem 1. *Let $E\left(T_A^{AFL}\right)$ be any upper bound on the expected runtime of algorithm A established by the artificial fitness levels method. Then $E\left(T^{AFL}_{(1+1)\ \textit{Fast-IA}_>}\right) \leq E\left(T^{AFL}_{(1+1)\ RLS_k}\right) \cdot k/\gamma \cdot O(1+\gamma \log n)$. Moreover, for the special case of $k = 1$, $E\left(T^{AFL}_{(1+1)\ \textit{Fast-IA}_\geq}\right) \leq E\left(T^{AFL}_{(1+1)\ RLS_1}\right) \cdot O(1+\gamma \log n)$ also holds.*

Proof. The (1+1) Fast-IA selects which k bits will be flipped first with the same distribution as (1+1) RLS$_k$ (uniformly at random without replacement) and evaluates the solution with probability γ/k if $k < n/2$ and with a probability greater than γ/k otherwise. Thus, from any initial solution, the (1+1) Fast-IA can improve at least with the same probability as (1+1) RLS$_k$ multiplied by γ/k if it is not stopped before the kth mutation step. With strict selection, it is guaranteed that the kth bitflip will happen at each iteration unless an improvement occurs. For the $k = 1$ case, the hypermutation cannot be stopped by a prior mutation step and the evaluation probability is always $1/e$. Pessimistically, we assume that if the exact k bits are not flipped, then the algorithm will not improve and all the fitness evaluations of the current mutation are wasted. The factor $O(1 + \gamma \log n)$ comes from Lemma 1. □

Theorem 3. *The expected runtime of the (1+1) Fast-IA to optimise* JUMP$_d$ *and* CLIFF$_d$ *is $O\left((d/\gamma) \cdot (1+\gamma \log n) \cdot \binom{n}{d}\right)$.*

Proof. According to Corollary 1, the time to sample a solution with $n-d$ 1-bits is at most $O\left(n \log n \left(1+\gamma \log n\right)\right)$ because the function behaves as ONEMAX for solutions with less than $n-d$ 1-bits. The Hamming distance of locally optimal points to the global optimum is d, thus, the probability of reaching the global optimum at the d_{th} mutation step is $\binom{n}{d}^{-1}$ while the probability of evaluating is γ/d. Using Lemma 1, we bound the total expected time to optimise JUMP and CLIFF as $E(T) \leq O(d/\gamma) \cdot O(1+\gamma \log n) \cdot \binom{n}{d}$. □

Theorem 2. *The expected runtime of the (1+1) Fast-IA to optimise* TRAP *is $\Theta(n \log n\,(1+\gamma \log n))$.*

Proof. According to Corollary 1 we can conclude that the current individual will reach 1^n in $O(n \log n \cdot (1 + \gamma \log n))$ steps in expectation. The global optimum is found in a single step with probability $1/e$ by evaluating after flipping every bit for which the number of additional fitness evaluations is $O(1 + \gamma \log n)$ in expectation. □

Theorem 4. *The Fast Opt-IA needs $O(\tau\mu + \mu n^{5/2} \log n)$ fitness function evaluations in expectation to optimise* HIDDENPATH *with $\mu = O(\log n)$, $dup = 1$, $1/(4 \ln n) \geq \gamma = \Omega(1/\log n)$ and $\tau = \Omega(n \log^2 n)$.*

Proof. We follow similar arguments to those of the proof of Theorem 11 in [17] for Opt-IA optimising HiddenPath. During the analysis, we call a non-SP solution which has i 0-bits an S_i solution for simplicity.

An S_{n-1} solution will be found in $1/e \cdot O(n \log n)$ generations by hill-climbing the ZeroMax part of the function. This individual creates and evaluates another S_{n-1} solution with probability at least $\gamma/2n$ (i.e., with probability $1/n$ the 1-bit is flipped and then any other bit is flipped with probability 1, and the solution will be evaluated with probability $\gamma/2$ after the second bit flip). After at most $\mu \cdot O(n)$ generations in expectation the whole population will consist only of S_{n-1} solutions since this solution is chosen as the parent with probability at least $1/\mu$ in each generation and then it is sufficient to flip the 1-bit in the first mutation step and then any 0-bit in the second mutation step to accept the offspring. Considering that the probability of producing two S_{n-1} in one generation is $\binom{\mu}{2} \cdot O(1/n) \cdot O(1/n) = O(\log^2 n/n^2)$, with probability at least $1 - o(1)$ we see at most one new S_{n-1} per generation for $o(n^2/\log^2 n)$ generations. Now, we can apply Lemma 3 of [11] to say that in $O(\mu^3 \cdot n)$ generations in expectation, the whole population reaches the same age while on the local optimum. After at most τ generations, with probability $(1-1/(\mu+1))^{2\mu-1} \cdot 1/(\mu+1)$, $\mu - 1$ b-cells die and one b-cell survives. In the following generation, while $\mu - 1$ randomly initialised b-cells are added instead of the dead b-cells, the survived b-cell creates an S_1 solution and evaluates it with probability $(1 - O(1/n))(1/e) = \Omega(1)$ for P-hype$_{FCM}$ by flipping all bits and evaluating the last bitstring. For P-hype$_{BM}$ it is necessary that only the final solution is evaluated. The expected number of evaluations between mutation steps two and $n-1$ is $2\sum_{i=2}^{n/2} \gamma/i \leq 1/2$ since $\gamma \leq 1/(4 \ln n)$, and the probability that there is at least one evaluation is at most $1/2$ by Markov's inequality. Thus, with probability $(1-1/e)(1/2)(1/e) = \Omega(1)$ only the complementary bitstring is evaluated and added to the population. In the following generation, this b-cell finds an S_5 solution by flipping at most six bits and evaluating it with probability at least $\gamma/6$. This individual will be added to the population with its age set to zero if the complementary bitstring (S_{n-1}) is not evaluated (with probability $(1-1/e)$). In the same generation the S_1 solution dies with probability $1/2$ due to ageing.

Next, we show that the S_5 solutions will take over the population, and the first point of SP will be found before any S_{n-1} is found. An S_5 creates an S_{n-5} individual and an S_{n-5} individual creates an S_5 individual with constant probability by evaluating complementary bitstrings. Thus, it takes $O(1)$ generations until the number of S_5 and S_{n-5} individuals in the population doubles. Since the increase in the total number of S_5 and S_{n-5} increases exponentially in expectation, in $O(\log \mu) = O(\log \log n)$ generations the population is taken over by them. After the take-over since each S_{n-5} solution creates a S_5 solution with constant probability, in the following $O(1)$ generations in expectation each S_{n-5} creates an S_5 solution which have higher fitness value than their parents and replace them in the population. Overall, S_5 solutions take over the population in $O(\log \log n)$ generations in expectation.

For S_5, HiddenPath has a gradient towards the SP which favors solutions with more 0-bits in the first 5 bit positions. Every improvement on the gradient takes $O(2/\gamma \cdot n^2)$ generations in expectation since it is enough to flip a a precise 1-bit and a 0-bit in the worst case. Considering that there are five different fitness values on the gradient, in $O(5 \cdot 2 \cdot n^2/\gamma) = O(n^2/\gamma)$ generations in expectation the first point of the SP will be found. Applying Markov's inequality, this time will not exceed $O(n^{5/2}/\gamma)$ with probability at least $1 - 1/\sqrt{n}$.

Now we go back to the probability of finding a locally optimal point before finding an SP point. Due to the symmetry of the hypermutation operator probability of creating an S_{n-1} solution from an S_5 solution is identical to create an S_{n-1} solution from and S_{n-5} solutions. The probability of increasing the number of 0-bits by k given that the initial number of 0-bits is k, is at most $(2i/n)^k$ due to the Ballot theorem since each improvement reinitialises a new ballot game with higher disadvantage (see the proof of Theorem 7 for a more detailed argument). Thus, the probability that a local optimal solution is sampled is $O(n^{-4})$. The probability that such an event never happens before finding SP is $1 - o(1)$. After finding SP, in $O(n \log n)$ generations in expectations the global optimum will be found at the end of the SP. The probability of finding any locally optimal point from SP is at most $O(1/n^4)$, hence this event would not happen before reaching the global optimum with probability $1 - o(1)$. Overall, the runtime is dominated by $O(\tau + n^{5/2}/\gamma)$ which give us $O((\tau + n^{5/2}/\gamma) \cdot \mu(1 + \gamma \log n))$ as the expected number of fitness evaluations due to Lemma 1. Since $\Omega(1/\log n) = \gamma \leq 1/(2 \ln n)$, the upper bound reduces to $O((\tau + \mu n^{5/2} \log n)$. □

Lemma 2. *The probability that a solution with at least $(n/2) + |a| + b$ 1-bits, where $-n/2 \leq a \leq n/2$ and $b < n/2$, is sampled at mutation step k, given the initial solution has $(n/2) + a$ 1-bits, is bounded above by $e^{-\frac{b^2}{k}}$.*

Proof. For the input bitstring x, let the multiset of weights $C := \{c_i | i \in [n]\}$ be defined as $c_i := (-1)^{x_i}$ (i.e., $c_i = -1$ if $x_i = 1$ and $c_i = 1$ if $x_i = 0$). Thus, for permutation π of bit-flips over $[n]$, the number of 1-bits after the kth mutation step is $\text{OneMax}(x) + \sum_{j=1}^{k} c_{\pi_j}$ since flipping the position i implies that the number of 1-bits changes by c_i. Let $\bar{X} := (1/k) \sum_{j=1}^{k} c_{\pi_j}$ be the sample mean, $\bar{\mu} := (1/n) \sum_{j=1}^{n} c_i$ the population mean. The number of 1-bits which incur the weight of -1 when flipped is at least $n/2 + a$. Thus, $\bar{\mu} \leq (1/n)\left(-(n/2) - a + (n/2) - a\right) = -2a/n$. To find a solution with at least $(n/2) + |a| + b$ 1-bits it is necessary that the sample mean $\bar{X}$ is at least $(|a| - a + b)/k$. Thus,

$$\bar{X} \geq \frac{|a| - a + b}{k} \implies \bar{X} - \bar{\mu} \geq \frac{|a| - a + b}{k} + \frac{2a}{n} \geq \frac{b}{k} + \frac{|a| - a}{k} + \frac{2a}{n} \geq \frac{b}{k}$$
$$\implies \sqrt{k}(\bar{X} - \bar{\mu}) \geq \frac{b}{\sqrt{k}}.$$

According to Theorem 8,

$$Pr\left\{\sqrt{k}(\bar{X}-\bar{\mu}) \geq \frac{b}{\sqrt{k}}\right\} \leq \exp\left(-\frac{2\left(\frac{b}{\sqrt{k}}\right)^2}{\left(1-\frac{k-1}{n}\right)(1-(-1))}\right) \leq \exp\left(-\frac{b^2}{k}\right).$$

□

Theorem 5. *Fast Opt-IA using P-hype$_{FCM}$ requires at least $2^{\Omega(n)}$ fitness function evaluations in expectation to find the optimum of* CLIFF$_d$ *for $d=(1-c)n/4$, where c is a constant $1>c>0$.*

Proof. Since each bit value in a solution initialised uniformly at random is equal to 1 with probability 1/2, the number of 1-bits in any initial solution is between $(n/2)-n^{3/5}$ and $(n/2)+n^{3/5}$ with overwhelmingly high probability due to Chernoff bounds. 1-bits. According to Lemma 2, the probability that an offspring with more than $a+n/10$ 1-bits is mutated from a parent with $a \geq (n/2)-n^{3/5}$ 1-bits is in the order of $2^{\Omega(n)}$ using the union bound. Therefore with overwhelmingly high probability we will not observe that a solution with less than $n-d-n/10$ 1-bits having an offspring with more than $n-d$ 1-bits. However, solutions with $n-d \geq\ b \geq n-d-n/10$ 1-bits, have higher fitness than post-cliff solutions with less than $b+d$ 1-bits. Thus, according to Lemma 2 the probability that an acceptable solution is obtained is in the order of $2^{-\Omega(d)}=2^{-\Omega(n)}$ again using union bound over n mutation steps.

Theorem 6. *Fast Opt-IA using P-hype$_{BM}$ with $\mu=1$, $dup=1$, $\gamma=1/(n\log^2 n)$ and $\tau=\Theta(n\log n)$ needs $O(n\log n)$ fitness function evaluations in expectation to optimise* CLIFF *with any linear $d \leq n/4-\epsilon$ for an small constant ϵ.*

Proof. Since $\gamma=1/(n\log^2 n)$, the expected number of fitness function evaluations per iteration $O(1+\gamma\log n)$ (see Lemma 1), is in the order of $\Theta(1)$. On the first ONEMAX slope, the algorithm improves by the first bit flip with probability at least $(n-d)/n=\Theta(1)$ and then evaluates this solution with probability $p_1=1/e=\Theta(1)$. This implies that the local optimum is found in $O(n)$ fitness evaluations in expectation after initialisation.

A solution at the local optimum can only improve by finding the unique globally optimum solution, which requires the hypermutation to flip precisely d 0-bits in the first d mutation steps which occurs with probability $\binom{n}{d}^{-1}$. We pessimistically assume that this direct jump never happens and assume that once a solution at the local optimum is added to the population, it reaches age τ in some iteration t_0. We consider the chain of events that starts at t_0 by **1)** the addition of a solution with $(n-d+1)$ 1-bits locally optimal to the population (with probability $(1/e)\cdot(n-d)/n$), **2)** the deletion of the locally optimal solution due to ageing with probability 1/2, **3)** the survival of the post-cliff solution with probability 1/2 and in iteration t_0+1, **4)** improvement of the post-cliff solution fitness function by hypermutation, which happens with a constant probability and effectively resets the new solution's age to zero. If all of these four events occur consecutively (which happens with constant probability), the algorithm can start climbing the second OneMax slope with local moves (i.e., by considering only the first mutation steps) which are evaluated with constant probability. Then, the Cliff function is optimised in $O(n \log n)$ function evaluations like OneMax unless a pre-cliff solution replaces the current individual. The rest of our analysis will focus on the probability that a pre-cliff solution is sampled and evaluated given that the algorithm has a post-cliff solution with age zero at iteration $t_0 + 1$.

If the current solution is a post-cliff solution, then the final bitstring sampled by the hypermutation has a worse fitness level than the current individual. The probability that P-hype$_{BM}$ evaluates at least one solution between mutation steps two and $n-1$ (event $\mathcal{E}_{nv}$), is upper bounded by $\sum_{n-1}^{i=2} \gamma/i < 2\gamma \cdot \log n = 2/(n \log n)$. We consider the $O(n \log n)$ generations until a post-cliff solution with age zero reaches the global optimum. The probability that event $\mathcal{E}_{nv}$ never occurs in any iteration until the optimum is found is at least $(1-2/(n\log n))^{O(n \log n)} = e^{-O(1)} = \Omega(1)$, a constant probability. Thus, every time we create a post-cliff solution with age zero, there is at least a constant probability that the global optimum is reached before any solution that is not sampled at the first or the last mutation step gets evaluated. The first mutation step cannot yield a pre-cliff solution, and the last mutation step cannot yield a solution with better fitness value. Thus, with a constant probability the post-cliff solution finds the optimum. If it fails to do so (i.e., a pre-cliff solution takes over as the current solution or a necessary event does not occur at iteration $t_0 + 1$), then in at most $O(n \log n)$ iterations another chance to create a post-cliff solution comes up and the process is repeated. In expectation, a constant number of trials will be necessary until the optimum is found and since each trial takes $O(n \log n)$ fitness function evaluations, thus our claim follows. $\square$

Theorem 7. *At least $2^{\Omega(n)}$ fitness function evaluations in expectation are executed before the Fast Opt-IA using P-hype$_{BM}$ with $\gamma = \Omega(1/\log n)$ finds the optimum of* Cliff$_d$ *for $d = (1-c)n/4$, where c is a constant $1 > c > 0$.*

Proof. Consider Fast Opt-IA with a current solution having more than $n - d$ (i.e., post-cliff) and less than $n - d + 2\sqrt{n}$ 1-bits. We will show that with overwhelmingly high probability, P-hype$_{BM}$ will yield a solution with less than $n-d$ (i.e., pre-cliff) and more than $n-2d+2\sqrt{n}$ 1-bits before the initial individual is mutated into a solution with more than $n - d + 2\sqrt{n}$ 1-bits. This observation will imply that a pre-cliff solution with better fitness will replace the post-cliff solution before the post-cliff solution is mutated into a globally optimal solution. We will then show that it is also exponentially unlikely that any pre-cliff solution mutates into a solution with more than $n-d+\sqrt{n}$ 1-bits to complete our proof.

We will now provide a lower bound on the probability that P-hype$_{BM}$ with post-cliff input solution x yields a pre-cliff solution with higher fitness value than x.

We will start by determining the earliest mutation step r_{min}, that a pre-cliff solution with worse fitness than x can be sampled. For any post-cliff solution x, $\textsc{Cliff}_d(x) = \textsc{OneMax}(x) - d + (1/2)$, and any pre-cliff solution y with $\textsc{OneMax}(x) - d + 1$ 1-bits has a higher fitness than x. We obtain the rough bound of $r_{min} \geq d - 2\sqrt{n}$ by considering the worst-case event that P-hype$_{BM}$ picks d 1-bits to flip consecutively.

Let $\ell(x)$ denote the number of extra 1-bits a post-cliff solution has in comparison to a locally optimal solution (i.e, $\textsc{OneMax}(x) = n - d + \ell(x)$). Next, we will use Serfling's bound to show that with a constant probability P-hype$_{BM}$ will find a pre-cliff solution before $3\ell(x)$ mutation steps and it will keep sampling pre-cliff solutions until r_{min}.

For the input bitstring of P-hype$_{BM}$, x, let the multiset of weights $C := \{c_i | i \in [n]\}$ be defined as $c_i := (-1)^{x_i}$ (i.e., $c_i = -1$ if $x_i = 1$ and $c_i = 1$ if $x_i = 0$). Thus, for a permutation π of bit-flips over $[n]$, the number of 1-bits after the kth mutation step is $\textsc{OneMax}(x) + \sum_{j=1}^{k} c_{\pi_j}$ since flipping the position i implies that the number of 1-bits changes by c_i.

Let $\bar{\mu} := (1/n)\sum_{j=1}^{n} c_i$ be the population mean of C and $\bar{X} := (1/3\ell(x))\sum_{j=1}^{3\ell(x)} c_{\pi_j}$ the sample mean. Since the $\textsc{Cliff}$ parameter d is less than $n/4$,

$$\bar{\mu} \leq (1/n)\left((-3n/4) + (n/4)\right) = -1/2$$

. In order to have a solution with at least $n-d+1$ 1-bits at mutation step $3\ell(x)$, the following must hold:

$$3\ell(x)\bar{X} \geq -\ell(x) \iff \bar{X} \geq -\frac{1}{3}$$
$$\implies \bar{X} - \bar{\mu} \geq -\frac{1}{3} + \frac{1}{2} = \frac{1}{6} \iff \sqrt{3\ell(x)}\left(\bar{X} - \bar{\mu}\right) \geq \frac{\sqrt{3\ell(x)}}{6}.$$

The probability that a pre-cliff solution will not be found in mutation step $3\ell(x)$ follows from Theorem 8, with sample mean $\bar{X}$, population mean $\bar{\mu}$, sample size $3\ell(x)$, population size n, $c_{min} = -1$ and $c_{max} = 1$.

$$Pr\left\{\sqrt{3\ell(x)}\left(\bar{X} - \bar{\mu}\right) \geq \frac{\sqrt{3\ell(x)}}{6}\right\} \leq \exp\left(-\frac{2\left(\frac{\sqrt{3\ell(x)}}{6}\right)^2}{\left(1 - \left(\frac{3\ell(x)-1}{n}\right)\right)(1-(-1))^2}\right)$$
$$\leq e^{-\Omega(\ell(x))}.$$

Thus, with probability $(1-e^{-\Omega(\ell(x))})$, we will sample the first pre-cliff solution after $3\ell(x)$ mutation steps. We focus our attention on post-cliff solutions with $1 \leq \ell(x) \leq 2\sqrt{n}$ and can conclude that for such solutions the above probability is in the order of $\Omega(1)$. Since the number of 0-bits changes by one at every mutation step, the event of finding a solution with at most $n-d$ bits implies that at some point a solution with exactly $n-d$ 1-bits has been sampled. Let $k_0 \leq 3\ell(x)$ be the mutation step where a locally optimum solution is found for the first time. Due to the Ballot theorem the probability that a solution with more than $n-d$ 1-bits is sampled after k_0 is at most $2d/n \leq 1/2$. So, with probability at least $1/2$, the P-hype$_{BM}$ will keep sampling pre-cliff solutions until $r_{min} \leq d - 2\sqrt{n} = \Omega(n)$. We will now consider the probability that at least one of the solutions sampled between k_0 and r_{min} is evaluated. Since the evaluation decisions are taken independently from each other the probability that none of the solutions are evaluated is

$$\prod_{i=k_0}^{r_{min}} \left(1-\frac{\gamma}{i}\right) \leq \prod_{i=3\ell(x)}^{r_{min}} \left(1-\frac{\gamma}{i}\right) \leq \prod_{i=6\sqrt{n}}^{r_{min}} \left(1-\frac{\gamma}{i}\right) \leq \prod_{i=6\sqrt{n}}^{r_{min}} \left(1-\frac{1}{(c_1 \log n) i}\right)$$

for some constant c_1 since $\gamma = \Omega(1/\log n)$. We will separate this product into $\lfloor \log{(r_{min}/6\sqrt{n})} \rfloor$ smaller products and show that each smaller product can be bounded from above by $e^{-1/(2c_1 \log)}$. The first subset contains the factors with indices $i \in \{(r_{min}/2)+1, \dots, r_{min}\}$, the second set $i \in \{(r_{min}/4)+1, \dots, r_{min}/2\}$ and jth set (for any $j \in [\lfloor \log{(r_{min}/6\sqrt{n})} \rfloor]$) $i \in \{r_{min}2^{-j}+1, \dots, r_{min}2^{-j+1}\}$. If some indices are not covered by these sets due to the floor operator, we will ignore them since they can only make the final product smaller. Note that we assume any logarithm's base is two unless it is specified otherwise.

$$\begin{aligned}
&\prod_{j=1}^{\lfloor \log{(r_{min}/6\sqrt{n})} \rfloor} \prod_{i=r_{min}2^{-j}+1}^{r_{min}2^{-j+1}} \left(1-\frac{1}{(c_1 \log n) i}\right) \\
&\leq \prod_{j=1}^{\lfloor \log{(r_{min}/6\sqrt{n})} \rfloor} \left(1-\frac{2^{j-1}}{(c_1 \log n) r_{min}}\right)^{r_{min}2^{-j}} \leq \prod_{j=1}^{\lfloor \log{(r_{min}/6\sqrt{n})} \rfloor} e^{-1/(2c_1 \log n)} \\
&\leq e^{-\lfloor \log{(r_{min}/6\sqrt{n})} \rfloor/(2c_1 \log n)} = e^{-\Omega(1)}
\end{aligned}$$

where in the second line we made use of the inequality $(1-cn^{-1})^n \leq e^{-c}$ and in the final line our previous observation that $r_{min} = \Omega(n)$. This implies that at least one of the sampled pre-cliff individuals will be evaluated at least with constant probability. At this point we have established that a pre-cliff solution will be added to the population with constant probability if the initial post-cliff solution x has a distance between $\sqrt{n}$ and $2\sqrt{n}$ to the local optima.

Let $\mathcal{E}_{i,k}$ for $k > 1$ denote the event that hypermutation samples a solution with $i-k$ 0-bits given that the initial solution has i 0-bits. Since the number of 0-bits change by one at every mutation, $\mathcal{E}_{i,k}$ implies $\mathcal{E}_{i,k-1}$. In particular, for $\mathcal{E}_{i,k}$ to happen first $\mathcal{E}_{i,k-1}$ must happen and then another improvement must be found. Given $\mathcal{E}_{i,k-1}$, the probability that a new improvement is found is less than $2i/n$ because of two reasons. First, the number of 0-bits has decreased with respect to the initial solution and second, there are fewer solutions to be sampled before P-hype$_{BM}$ terminates. Thus, we can conclude: $\Pr\{\mathcal{E}_{i,k}\} \leq \Pr\{\mathcal{E}_{i,k-1}\} \frac{2i}{n} \leq \left(\frac{2i}{n}\right)^k$. This implies that it is exponentially unlikely that a pre-cliff solution is mutated into a solution with more than $n-d+\sqrt{n}$ 1-bits. Moreover, the probability that post-cliff solutions are improved by more than $n^{1/6}$ is less than $4^{-n/6}$, which implies that with overwhelmingly high probability it takes at least $n^{(1/2)-(1/6)} =$

$n^{1/3}$ iterations before a solution x with $\ell(x) < \sqrt{n}$ is mutated into a solution x' with $\ell(x') > 2\sqrt{n}$. Since we established that a pre-cliff solution is evaluated with constant probability at each iteration, we can conclude that at least one such individual is sampled in $n^{1/3}$ iterations with overwhelmingly high probability. Since the Fast Opt-IA cannot follow the post-cliff gradient to the optima with overwhelmingly high probability it relies on making the jump from local optima to global optima. Given an initial solution with $y \in \{(n/3), \ldots, n - d + 2\sqrt{n}\}$ 1-bits, the probability of jumping to the unique global optimum is $2^{\Omega(-n)}$ as well, thus our claim follows. $\square$