Fast Artificial Immune Systems

Dogan Corus(✉), Pietro S. Oliveto, and Donya Yazdani

Rigorous Research, University of Sheffield, Sheffield, UK
{d.corus,p.oliveto,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 bit-flip 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], the B-Cell algorithm [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

[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.

© Springer Nature Switzerland AG 2018
A. Auger et al. (Eds.): PPSN 2018, LNCS 11102, pp. 67–78, 2018.
https://doi.org/10.1007/978-3-319-99259-4_6

average 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 CHM 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 ith 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. Due to space limitations some proofs are omitted for this extended abstract[2].

2 Preliminaries

Static hypermutations with mutation potential using FCM (i.e., stop at the first constructive mutation) mutate $M = cn$ distinct bits for a constant $0 < c \leq 1$ and evaluate the fitness after each bit-flip [6]. If an improvement over the original solution is found before the Mth bit-flip, 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 bit-flip 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 ith bit has been flipped. The 'parabolic' probability distribution is defined as follows, where the parameter γ should be between $0 < \gamma \leq 2$ (Fig. 1):

[2] A complete version of the paper including all the proofs is available on arXiv [13].

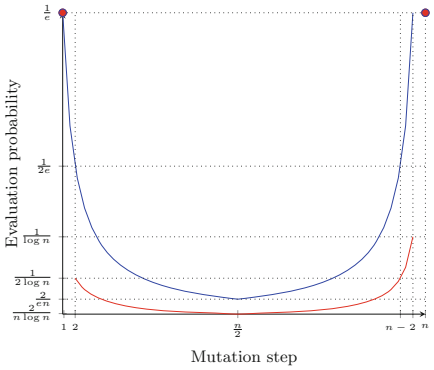

Fig. 1. The parabolic evaluation probabilities (1) for $\gamma = 1/\log n$ and $\gamma = 1/e$.

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

The lower the value of γ, the fewer the expected fitness function evaluations that occur in each hypermutation. 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 ith bit-flip 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 ith bit-flip 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 Sect. 4, we consider P-hype$_{BM}$ in the complete Opt-IA framework [4–6] 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

Initialise a population of μ b-cells, P, created uniformly at random;
for each $x \in P$ set $x^{age} = 0$.
while a global optimum is not found **do**
  for each $x \in P$ set $x^{age} = x^{age} + 1$;
  for *dup* times for each $x \in P$ **do**
    $y =$ P-hype(x);
    if $f(y) > f(x)$ **then** $y^{age} = 0$ **else** $y^{age} = x^{age}$;
    Add y to P^{hyp}.
  end for
  Add P^{hyp} to P, set $P^{hyp} = \emptyset$;
  for each $x \in P$ **if** $x^{age} \geq \tau$ **then** remove x with probability p_{die};
  if $|P| < \mu$ **then** add $\mu - |P|$ solutions to P with age zero generated uniformly at random;
  if $|P| > \mu$ **then** remove $|P| - \mu$ solutions with the lowest fitness from P breaking ties uniformly at random.
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 ith 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 [14] 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, $E[X] = E\left[\sum_{i=1}^{n} Y_i\right] = \sum_{i=1}^{n} Pr\{Y_i = 1\} = \frac{1}{e} + \frac{1}{e} + 2\sum_{i=2}^{n/2} \gamma \frac{1}{i} \leq \frac{2}{e} + 2\gamma(\ln n/2 - 1)$. □

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 Sect. 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 and cliff sizes both AISs are much faster than the $(1+1)$ EA.

Function	$(1+1)$ EA	$(1+1)$ IA$^{\text{hyp}}$	$(1+1)$ Fast-IA
OneMax	$\Theta(n \log n)$ [15]	$\Theta(n^2 \log n)$ [6]	$\Theta\left(n \log n \left(1+\gamma \log n\right)\right)$
LeadingOnes	$\Theta(n^2)$ [15]	$\Theta(n^3)$ [6]	$\Theta\left(n^2 \left(1+\gamma \log n\right)\right)$
Trap	$\Theta(n^n)$ [15]	$\Theta(n^2 \log n)$ [6]	$\Theta\left(n \log n \left(1+\gamma \log n\right)\right)$
$\text{Jump}_{d>1}$	$\Theta(n^d)$ [15]	$O(n\binom{n}{d})$ [6]	$O\left((d/\gamma) \cdot (1+\gamma \log n) \cdot \binom{n}{d}\right)$
$\text{Cliff}_{d>1}$	$\Theta(n^d)$ [16]	$O(n\binom{n}{d})$ [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 Theorems 3.3 and 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 $E\left(T^{AFL}_{(1+1)\ Fast\text{-}IA_{>}}\right) \leq E\left(T^{AFL}_{(1+1)\ RLS_k}\right) \cdot k/\gamma \cdot O(1+\gamma \log n)$. Moreover, for the special case of $k=1$, $E\left(T^{AFL}_{(1+1)\ Fast\text{-}IA_{\geq}}\right) \leq E\left(T^{AFL}_{(1+1)\ RLS_1}\right) \cdot O(1+\gamma \log n)$ 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* OneMax$(x) := \sum_{i=1}^{n} x_i$ *and* 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, EAs based on SBM require exponential runtime with overwhelming probability [17]. 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$^{\text{hyp}}$ 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$^{\text{hyp}}$ 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 Jump$_d$ and 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* Jump$_d$ *and* 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$^{\text{hyp}}$) 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.

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 a worsening; otherwise the second gradient is not accessible. To prove our upper bound, we can follow the same proof strategy in [18], 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 $n/\log n$ speed-up due to P-hype.

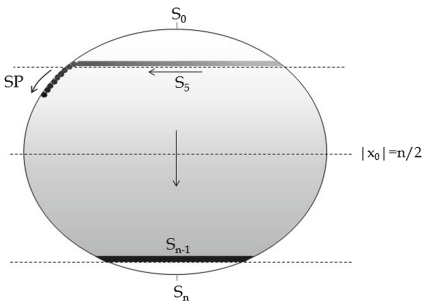

Fig. 2. HiddenPath [6]

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 [18]. 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 positive 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$_{FCM}$, that, instead of stopping the hypermutation at the first constructive mutation, will execute all n mutation steps, evaluate 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* Cliff *with any linear $d \leq n/4 - \epsilon$ 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 [19–22].

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 speed-ups in the exploitation phase.

The main modification that allows to achieve the presented improvements is to sample the solution after the ith bit-flip 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 (e.g., 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 [23,24]. Furthermore, the performance of the proposed operators should be evaluated for classical combinatorial optimisation problems and real-world applications.

References

1. Burnet, F.M.: The Clonal Selection Theory of Acquired Immunity. Cambridge University Press, Cambridge (1959)
2. de Castro, L.N., Von Zuben, F.J.: Learning and optimization using the clonal selection principle. IEEE Trans. Evol. Comput. **6**(3), 239–251 (2002)
3. Kelsey, J., Timmis, J.: Immune inspired somatic contiguous hypermutation for function optimisation. In: Cantú-Paz, E., et al. (eds.) GECCO 2003. LNCS, vol. 2723, pp. 207–218. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45105-6_26
4. Cutello, V., Nicosia, G., Pavone, M., Timmis, J.: An immune algorithm for protein structure prediction on lattice models. IEEE Trans. Evol. Comput. **11**(1), 101–117 (2007)
5. Cutello, V., Nicosia, G., Pavone, M.: A hybrid immune algorithm with information gain for the graph coloring problem. In: Cantú-Paz, E. (ed.) GECCO 2003. LNCS, vol. 2723, pp. 171–182. Springer, Heidelberg (2003). https://doi.org/10.1007/3-540-45105-6_23

6. Corus, D., Oliveto, P.S., Yazdani, D.: On the runtime analysis of the Opt-IA artificial immune system. In: Proceedings of the GECCO 2017, pp. 83–90 (2017)
7. Jansen, T., Zarges, C.: Analyzing different variants of immune inspired somatic contiguous hypermutations. Theor. Comput. Sci. **412**(6), 517–533 (2011)
8. Jansen, T., Zarges, C.: Computing longest common subsequences with the B-Cell algorithm. In: Coello Coello, C.A., Greensmith, J., Krasnogor, N., Liò, P., Nicosia, G., Pavone, M. (eds.) ICARIS 2012. LNCS, vol. 7597, pp. 111–124. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33757-4_9
9. Jansen, T., Oliveto, P.S., Zarges, C.: On the analysis of the immune-inspired B-Cell algorithm for the vertex cover problem. In: Liò, P., Nicosia, G., Stibor, T. (eds.) ICARIS 2011. LNCS, vol. 6825, pp. 117–131. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-22371-6_13
10. Corus, D., He, J., Jansen, T., Oliveto, P.S., Sudholt, D., Zarges, C.: On easiest functions for mutation operators in bio-inspired optimisation. Algorithmica **78**(2), 714–740 (2016)
11. Oliveto, P.S., Sudholt, D.: On the runtime analysis of stochastic ageing mechanisms. In: Proceedings of the GECCO 2014, pp. 113–120 (2014)
12. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**(4), 623–642 (2012)
13. Corus, D., Oliveto, P.S., Yazdani, D.: Fast artificial immune systems. ArXiv e-prints (2018). http://arxiv.org/abs/1806.00299
14. Mitzenmacher, M., Upfal, E.: Probability and Computing: Randomized Algorithms and Probabilistic Analysis. Cambridge University Press, Cambridge (2005)
15. Droste, S., Jansen, T., Wegener, I.: On the analysis of the (1 + 1) evolutionary algorithm. Theor. Comput. Sci. **276**(1–2), 51–81 (2002)
16. Paixão, T., Heredia, J.P., Sudholt, D., Trubenová, B.: Towards a runtime comparison of natural and artificial evolution. Algorithmica **78**(2), 681–713 (2017)
17. Oliveto, P.S., Yao, X.: Runtime analysis of evolutionary algorithms for discrete optimization. In: Auger, A., Doerr, B. (eds.) Theory of Randomized Search Heuristics, pp. 21–52. World Scientific (2011)
18. Corus, D., Oliveto, P.S., Yazdani, D.: When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. ArXiv e-prints (2018). http://arxiv.org/abs/1804.01314
19. Oliveto, P.S., Lehre, P.K., Neumann, F.: Theoretical analysis of rank-based mutation-combining exploration and exploitation. In: Proceedings of the CEC 2009, pp. 1455–1462 (2009)
20. Doerr, B., Le, H.P., Makhmara, R., Nguyen, T.D.: Fast genetic algorithms. In: Proceedings of the GECCO 2017, pp. 777–784 (2017)
21. Corus, D., Oliveto, P.S.: Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. IEEE Trans. Evol. Comput. (2017)
22. Dang, D.-C., et al.: Emergence of diversity and its benefits for crossover in genetic algorithms. IEEE Trans. Evol. Comput. (2017, to appear)
23. Doerr, B., Lissovoi, A., Oliveto, P.S., Warwicker, J.A.: On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In: Proceedings of the GECCO 2018. ACM (2018, to appear)
24. Doerr, B., Doerr, C.: Optimal static and self-adjusting parameter choices for the $(1 + (\lambda, \lambda))$ genetic algorithm. Algorithmica **80**, 1658–1709 (2018)