Fast Immune System Inspired Hypermutation Operators for Combinatorial Optimisation

Dogan Corus, Pietro S. Oliveto, and Donya Yazdani
dogan.corus@khas.edu.tr, p.oliveto@sheffield.ac.uk, d.yazdani@aber.ac.uk

***Abstract*—Various studies have shown that immune system inspired hypermutation operators can allow artificial immune systems (AIS) to 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. This 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. The stochastic distribution also allows the removal of the FCM mechanism altogether as originally desired in the design of the HMP operators. We rigorously prove the effectiveness of the proposed operators for all the benchmark functions where the performance of HMP is rigorously understood in the literature. We validate the gained insights to show linear speed-ups for the identification of high quality approximate solutions to classical NP-Hard problems from combinatorial optimisation. We then show the superiority of the HMP operators to the traditional ones in an analysis of the complete standard Opt-IA AIS, where the stochastic evaluation scheme allows HMP and ageing operators to work in harmony. Through a comparative performance study of other 'fast mutation' operators from the literature, we conclude that a power-law distribution for the parabolic evaluation scheme is the best compromise in black-box scenarios where little problem knowledge is available.**

***Index Terms*—Artificial immune systems, Hypermutation, Runtime analysis**

I. Introduction

Several artificial immune systems (AISs) 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] are the most popular. As they are all inspired by the immune system, a common feature of these algorithms is that they have particularly high mutation rates compared to more traditional evolutionary algorithms (EAs) which, inspired in turn by natural evolution, have traditionally used considerably lower mutation rates.

Dogan Corus is based at the Computer Engineering Department, Kadir Has University, Istanbul, Turkey.
Pietro S. Oliveto is with the Department of Computer Science, The University of Sheffield, Sheffield, UK.
Donya Yazdani is with the Advanced Reasoning Group, Department of Computer Science, Aberystwyth University, Aberystwyth, UK.

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 on average in a mutation. The *hypermutations with mutation potential* (HMP) used by Opt-IA also flip a linear number of bits. However, it has been proved that their basic originally proposed static version, where a linear number of bits are always flipped, cannot optimise efficiently any function with any polynomial number of optima [5]. On the other hand, much better performance has been shown in theory [5] and in practice [4] for the version that evaluates the fitness after each bit flip in the hypermutation and stops the process if an improving solution is found (i.e., static HMP with *stop at first constructive mutation* (FCM)).

Various studies have shown how these high mutation rates allow AISs 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 [6]. Similar effects have also been shown for instances of the longest common subsequence [7] and vertex cover [8] combinatorial optimisation problems with practical applications, where CHM efficiently escapes local optima while EAs (with and without crossover) are trapped for exponential time. Also, the HMP with FCM of Opt-IA has 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 [5]. Furthermore, their effectiveness at escaping from local optima has been shown to guarantee arbitrarily good constant approximations for the NP-Hard PARTITION problem while RLS (Randomised Local Search) and EAs may get stuck on bad approximations [9].

The efficiency on multimodal problems of these AISs comes at the expense of being considerably slower than EAs 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 unimodal benchmark functions. Indeed, it has recently been shown that CHM requires at least $\Omega(n^2)$ function evaluations to optimise any function since

[1]A parameter may be used to define the probability that each bit in the region actually flips.

its expected runtime for its easiest function is $\Theta(n^2)$ [10]. Another 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 bit-string. In this sense, the *unbiased* HMP operators used by Opt-IA are easier and more convenient to apply as their performance does not depend on the encoding order of the bit positions. However, the static HMP operator with FCM has also been proven to have runtimes of respectively $\Theta(n^2 \log n)$ expected fitness evaluations for ONEMAX and $\Theta(n^3)$ for LEADINGONES. Recently, speed-ups in the exploitation phase have been shown for the Inversely Proportional HMP variant (INV HMP), that aims to decrease the mutation rate as the local and global optima are approached [11]. On one hand, while faster, INV HMP operators are still asymptotically slower than RLS and EAs for easy hillclimbing problems such as ONEMAX and LEADINGONES. On the other hand, the speed-ups at hillclimbing are achieved at the expense of losing their power at escaping from local optima via mutation. Since the mutation rates are lowest on local optima, it is unlikely that the INV HMP operator can escape quickly via hypermutation.

In this paper, we propose a modification to the static 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 HMP with FCM requires, we propose to evaluate the fitness 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 (i.e., $\binom{n}{i}^{-1}$) 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 the solution after each bit flip following a parabolic distribution such that the probability of evaluating after the ith bit flip decreases as i approaches $n/2$ and then increases again. We call the resulting operator FCM$_\gamma$ and embed it in an algorithm called Fast $(1+1)$ IA$_\gamma$.

We rigorously prove that the Fast $(1+1)$ IA$_\gamma$ locates local optima asymptotically as quickly as random local search (RLS) for any function where the expected runtime of RLS can be proven using the standard artificial fitness levels method (AFL). At the same time, the operator is still exponentially faster than EAs for the standard multimodal JUMP, CLIFF, and TRAP benchmark functions.

We also validate the insights gained from the analysis for benchmark functions on classical NP-Hard problems from combinatorial optimisation. We first derive a smaller upper bound compared to static HMP on the expected runtime required by the Fast $(1+1)$ IA$_\gamma$ to find arbitrarily good constant approximations to the PARTITION problem. This result is surprising because the proof requires mutations of approximately $n/2$ bits. This is exactly the range of mutations which is penalised by our proposed distribution. Nevertheless, the greater exploitative capabilities of the hypermutation operator lead to a linear factor smaller upper bound on the expected runtime because the time spent in the hillclimbing phases dominates the overall expected runtime. Thus, the utility of our modifications is proven on a problem with many real world applications. Recall that EAs using standard bit mutation (SBM) may get stuck on bad $4/3$ approximations for exponential time. We also rigorously prove linear speed-ups for the NP-Hard VERTEX COVER problem, compared to the static HMP operator. We show these both for identifying feasible solutions if a node representation is used, and to identify 2-approximations for edge-based representations.

We then evaluate the performance of the *fast* hypermutation operator using the parabolic evaluation distribution in the context of complete AISs. Indeed hypermutations with mutation potential are usually applied in conjunction with ageing operators in the standard Opt-IA AIS [4]. The power of ageing at escaping local optima has recently been enhanced by showing how, by accepting inferior solutions when stuck on local optima, it makes the difference between polynomial and exponential runtimes for the BALANCE function from dynamic optimisation [12]. For very difficult instances of CLIFF, where standard RLS and elitist EAs require exponential time, ageing even makes RLS asymptotically as fast as any unbiased mutation based algorithm can be on any function with unique optimum [13] i.e., by running in $O(n \ln n)$ expected time [5].

However, the power of ageing at escaping local optima is lost when it is used in combination with static HMP. In particular, the FCM mechanism does not allow the operator to return solutions of lower quality apart from the complementary bit-string, thus cancelling the advantages of ageing. Furthermore, the high mutation rates combined with FCM make the algorithm return to the previous local optimum with high probability. We show how these problems are naturally solved by our newly proposed operators that do not evaluate all bit flips in a hypermutation. We rigorously prove that the resulting algorithm, called Fast Opt-IA$_\gamma$, benefits from the modified operator showing that it allows the ageing operator to escape from local optima by accepting the lower quality solutions returned by the FCM$_\gamma$ operator when it does not find improvements. To achieve this, the evaluation probabilities after each bit flip have to be set to prohibitively low values such that the applied operator effectively does not mutate many bits anymore (i.e. it does not hypermutate; similarly to the INV HMP of [11] when it is located on the best found local optimum).

To address this problem, and to further evaluate the general performance of the proposed *fast* HMP operator, we perform a comparative analysis with other '*fast* mutation' operators that have recently appeared in the evolutionary computation literature [14]–[16]. The analysis leads to the conclusion that a parabolic power-law distribution is the best compromise for the *fast* hypermutation operator in black box scenarios where limited problem knowledge is available. Such a distribution allows a greater balance between large and small mutations. Hence, local optima may be escaped from, by performing large or small mutations to new basins of attraction that are either of better or of worse quality (i.e., due to ageing). We show that the obtained AISs perform asymptotically either at least as well, or better, than all the considered algorithms over the large range of unimodal and multimodal problems considered in this paper. Due to page restrictions the proofs are presented as supplementary material as well as a self-contained version.

II. AISs with Probabilistic Sampling Distributions

Hypermutations with mutation potential (HMP) differ from the standard bit mutations (SBM) used traditionally in evolutionary computation by flipping a linear number of distinct bits $M = cn$ for a constant $0 < c \leq 1$. It has been shown that in their basic static version, where they only evaluate the result of the M bit flips, they are inefficient at optimising any function with up to a polynomial number of optima [5]. In the *stop at the first constructive mutation* (FCM) variant they mutate *at most* $M = cn$ distinct bits (i.e., for this reason M is called the mutation *potential*). After each of the M bit-flips, they evaluate the fitness of the constructed solution. If an improvement over the original solution is found before the M-th bit-flip, then the operator stops and returns the improved solution [4]. 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. While they have been shown to be more efficient than SBM to escape from local optima, this performance comes at the expense of being up to a linear factor slower at hillclimbing in the exploitative phases of the optimisation process [5].

Therefore, we propose an alternative HMP operator using FCM, called FCM$_\gamma$ for simplicity, that only evaluates the fitness after each bit-flip with some probability. Since setting the HMP parameter to $c = 1$ (i.e., $M = n$) allows the operator to reach any point in the search space with positive probability, we will only consider this parameter setting throughout the paper as was also done in previous work [5], [17].

We propose the use of the following parabolic probability distribution depicted in Figure 1. Let p_i be the probability that the solution is evaluated after the i-th bit has been flipped. Then,

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

The parameter γ should be in $0 < \gamma \leq 1$ (however, any $0 < \gamma < 1/e$ is an efficient choice for the results we present).

The lower the value of γ, the fewer the expected fitness function evaluations that occur in each hypermutation. In particular, with a sufficiently small value for γ, 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. At the same time, it still flips many bits (i.e., it hypermutates) as desired. The hypermutation operator is formally defined as follows.

Definition 1 (FCM$_\gamma$). *The FCM$_\gamma$ operator 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. FCM$_\gamma$ 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.*

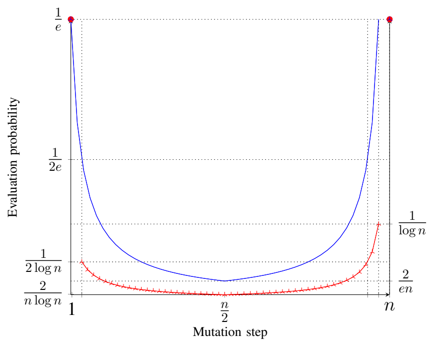

Fig. 1: The parabolic evaluation probabilities (1) for $\gamma = 1/\log n$ (starred) and $\gamma = 1/e$.

Algorithm 1 Fast $(1+1)$ IA$_\gamma$ for maximisation

```
1: Initialise $x$ u.a.r (uniformly at random).
2: while the termination criterion is not met do
3:     create offspring $y$ using FCM$_\gamma$;
4:     if $f(y) \geq f(x)$, then $x := y$;
5: end while
```

In the next section, we will prove the benefits of FCM$_\gamma$ over the standard HMP with FCM, when incorporated into a $(1+1)$ framework. We will refer to the algorithm as Fast $(1+1)$ IA$_\gamma$ to distinguish it from the standard $(1+1)$ IA which uses the traditional HMP operator i.e., that evaluates the fitness of the constructed solutions deterministically after each bit-flip of the hypermutation. Similar benefits may also be shown for population-based AISs but we will refrain to do so since populations do not lead to improved performance for the considered benchmark problems. The Fast $(1+1)$ IA$_\gamma$ is formally defined in Algorithm 1. It keeps a single individual in the population and uses FCM$_\gamma$ to perturb it in every iteration. If the offspring is not worse than its parent, then it replaces the parent for the next iteration; otherwise the parent is kept.

Traditional static FCM operators are not suited to be used in conjunction with ageing operators if the power of the latter at escaping local optima is to be exploited [5]. While ageing operators allow to exploit solutions of lower quality to escape from local optima, the traditional HMP with FCM returns a solution if it is an improvement or it always returns the complementary bit string (which is unlikely to be useful very often). However, this is not true for the above defined FCM$_\gamma$ variant. If no improvements are found, FCM$_\gamma$ returns the last evaluated solution, which is not necessarily the complementary bit string. Hence, the above operator has higher chances of being effective at escaping from local optima than traditional HMP with FCM by identifying a variety of new, potentially promising, basins of attraction. For sufficiently small values of the parameter γ, only one function evaluation per hypermutation is performed in expectation (although all bits will be flipped i.e., it hypermutates). Since FCM$_\gamma$ returns the last evaluated one, this solution will be returned by the operator as it is the only one it has encountered. Interestingly, this behaviour is similar to that of the traditional HMP operator without FCM that also evaluates one point per hypermutation and returns it. However, while the traditional version has been proven to have exponential expected runtime for any function with any polynomial number of optima [5], we will show in the following sections that the *fast* HMP can be very efficient. From this point of view, with appropriate parameter settings, FCM$_\gamma$ is a very effective way to perform hypermutations with mutation potential without FCM as originally desired [4].

We will analyse the FCM$_\gamma$ operator in a complete Opt-IA that uses cloning, hypermutation and ageing. The modified Opt-IA algorithm using FCM$_\gamma$, which we call Fast Opt-IA$_\gamma$, is depicted in Algorithm 2. It uses the *hybrid ageing* operator as in [5], [12], which allows the algorithm to escape from local optima. Hybrid ageing removes candidate solutions (i.e., b-cells) with probability p_{die} once they have reached an age threshold τ. After initialising a population of μ solutions with age zero (i.e., $\alpha(x_i) = 0$), the algorithm creates λ copies (i.e., clones) of each solution. The clones are all mutated by the hypermutation operator, creating a population of mutants called $P^{(hyp)}$. The mutants inherit the age of their parents if they do not improve the fitness; otherwise their age is set to zero. At the next step, all solutions with age greater equal τ are removed with probability p_{die}. If fewer than μ individuals survive ageing, then the population is filled up with new randomly generated individuals. Finally, the best μ solutions are chosen to form the population for the next generation. In Section V, we will prove the benefits of the Fast Opt-IA$_\gamma$ for all the unimodal and multimodal benchmark functions for which the performance of the Opt-IA with traditional static HMP has been proven in the literature.

As usual in evolutionary computation we will evaluate the performance of the algorithms by calculating the expected number of fitness function evaluations until the optimum (or an approximation for the NP-Hard problems) is identified (i.e. expected runtime). Hence, we do not specify any termination criterion for the evolutionary loops of the algorithms.

Algorithm 2 Fast Opt-IA$_\gamma$ for maximisation

```
1: Initialise $P := \{x_1, ..., x_\mu\}$, a population of $\mu$ solutions
   u.a.r and set $\alpha(x_i) := 0$ for $i := \{1, ...\mu\}$;
2: while the termination criterion is not met do
3:     for all $x \in P$ do
4:         set $\alpha(x) := \alpha(x) + 1$;
5:         copy $x$ $\lambda$ times and add the copies to $P^{(clo)}$;
6:     end for
7:     for all $x \in P^{(clo)}$ do
8:         create $y$ using FCM$_\gamma$;
9:         if $f(y) > f(x)$, then $\alpha(y) := 0$;
10:        else $\alpha(y) := \alpha(x)$;
11:        add $y$ to $P^{(hyp)}$;
12:    end for
13:    add $P^{(hyp)}$ to $P$, set $P^{(hyp)} := \emptyset$;
14:    with probability $p_{die} := 1 - \frac{1}{(\lambda+1)\cdot\mu}$, remove any $x_i \in$
       $P$ with $\alpha(x_i) \geq \tau$;
15:    if $|P| < \mu$, then add $\mu - |P|$ solutions to $P$ with age
       zero generated u.a.r;
16:    else if $|P| > \mu$, then remove $|P| - \mu$ solutions with the
       lowest fitness from $P$ breaking ties u.a.r;
17: end while
```

III. Artificial Fitness Levels for Fast Hypermutations

In [5], a mathematical methodology was devised that allows to convert upper bounds on the expected runtime of RLS into valid upper bounds on the expected runtime of the traditional static HMP operators. In this section, we will extend such methodology so that it can also be applied to the *fast* HMP operator introduced in this paper.

Artificial Fitness Levels (AFL) is a standard technique used in the theory of evolutionary computation to derive upper bounds on the expected runtime of $(1+1)$ evolutionary algorithms [18]–[20]. AFL divides the search space into m mutually exclusive partitions $A_1 \cdots, A_m$ such that all the points in A_i have smaller fitness than any point 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 $E(T) \leq \sum_{i=1}^{m-1} 1/p_i$.

RLS flips exactly 1 bit 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. The artificial fitness levels method for the traditional static HMP operator from [5] states that any upper bound on the expected runtime of RLS proven using the artificial fitness levels (AFL) method also holds for the $(1+1)$ IA multiplied by an additional factor of n (i.e., the algorithm is at most a linear factor slower than RLS for problems where the original upper bound is tight). The result was shown to be tight for some standard benchmark functions including OneMax and LeadingOnes. We will now extend the methodology to also hold for the *fast* HMP operator defined in the previous section by establishing a relationship between the upper bounds on the expected runtimes of RLS achieved via AFL and those of the Fast $(1+1)$ IA. However, these upper bounds will differ only by a factor of $O(1 + \gamma \log n)$ instead of n. Thus, for values of $\gamma = O(1/\log n)$, the upper bounds of the two algorithms are asymptotically the same, and the methodology will allow to prove a linear speed up for the *fast* HMP operator compared to traditional static HMP for the cases where the AFL methodology from [5] is tight.

We start our analysis by relating the expected number of fitness function evaluations to the expected number of fast hypermutation operations until an optimum is found. The lemma quantifies the number of expected fitness function evaluations performed by the two operators in one hypermutation.

Lemma 1. *Let T be the random variable denoting the number of applications of FCM$_\gamma$ with parameter $0 < \gamma < 1$ until the optimum is found. Then, the expected number of function evaluations in an FCM$_\gamma$ operation given that no improvement is found is in the order of $\Theta(1 + \gamma \log n)$. Moreover, the expected number of total function evaluations is at most*

$O(1+\gamma\log n)\cdot E[T]$.

In Lemma 1, the evaluation parameter 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 γ will not decrease the expected runtime. Nevertheless, in Section V 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 HMP with FCM while the number of wasted fitness function evaluations decreases from n to $O(1+\gamma\log n)$. We now present the main result of this section. The theorem applies to $(1+1)$ frameworks using the FCM$_\gamma$ as hypermutation operator.

Theorem 2. *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,*

$$E\left(T_{FCM_\gamma}\right)\leq E\left(T_{RLS}^{AFL}\right)\cdot O(1+\gamma\log n).$$

Apart from showing the efficiency of the Fast $(1+1)$ IA$_\gamma$, the theorem also allows to easily achieve upper bounds on the expected runtime of the algorithm by just analysing the simple RLS. For $\gamma=O(1/\log n)$, Theorem 2 implies the upper bounds of $O(n\log n)$ and $O(n^2)$ for classical benchmark functions $\textsc{OneMax}(x)=\sum_{i=1}^n x_i$ and $\textsc{LeadingOnes}(x)=\sum_{i=1}^n\prod_{j=1}^i x_j$ respectively [18]. These expected runtimes represent linear speed-ups compared to the (1+1) IA using the static HMP operators from the literature which have $\Theta(n^2\log n)$ and $\Theta(n^3)$ expected runtimes for OneMax and LeadingOnes respectively [5]. By providing ad-hoc lower bounds we show that the upper bounds on the expected runtimes provided by the AFL method are tight.

Corollary 3. *The expected runtimes of the Fast $(1+1)$ IA$_\gamma$ using FCM$_\gamma$ 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 $\Theta\left(n\log n\left(1+\gamma\log n\right)\right)$ and $\Theta(n^2\left(1+\gamma\log n\right))$. For $\gamma=O(1/\log n)$ these bounds reduce to $\Theta(n\log n)$ and $\Theta(n^2)$.*

IV. Fast Hypermutations for Standard Multimodal Benchmark Functions

In the previous section we showed that linear speed-ups compared to static HMP are achieved by the Fast $(1+1)$ IA$_\gamma$ for standard unimodal benchmark functions i.e., the algorithm is fast at exploitation for hill-climbing problems. In this section we will show that exponential speed-ups compared to the standard bit mutation operators used in traditional EAs are still achieved for standard multimodal benchmark functions i.e., the Fast HMP operators are also efficient at exploration.

We start by using the mathematical methodology derived in the previous section to show that the Fast $(1+1)$ IA$_\gamma$ is even faster than static HMP for the deceptive Trap function which is identical to OneMax except that the optimum is in 0^n. FCM$_\gamma$ samples the complementary bit-string with probability one if it cannot find any improvements. This behaviour allows it to be efficient for this deceptive function. Since n bits have to be flipped to reach the global optimum from the local optimum, EAs with SBM require exponential runtime with

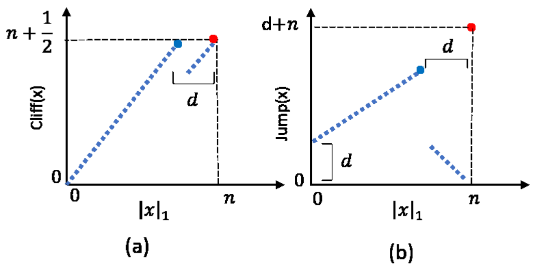

Fig. 2: (a) Cliff$_d$ and (b) Jump$_d$

overwhelming probability (w.o.p.)[2] [18]. By evaluating the sampled bit-strings stochastically, the Fast (1+1) IA$_\gamma$ provides up to a linear speed-up for small enough γ compared to the $(1+1)$ IA on Trap as well.

Theorem 4. *The expected runtime of the Fast $(1+1)$ IA$_\gamma$ for Trap is $O(n\log n\left(1+\gamma\log n\right))$.*

The results of the $(1+1)$ IA on Jump$_d$ and Cliff$_d$ functions [5] can also be adapted to the Fast $(1+1)$ IA$_\gamma$ in a straightforward manner.

Both Jump$_d$ and Cliff$_d$ have the same structure as OneMax for bit-strings 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$_d$ has a reversed OneMax slope creating a gradient towards $n-d$ while Cliff$_d$ has a slope heading toward 1^n, but the fitness values are penalised by an additive factor d. These functions are illustrated in Fig. 2. Since hypermutation operators have a higher probability of flipping multiple bits, the performance of static HMP on the Jump$_d$ and Cliff$_d$ functions is superior to that of the SBM used by traditional EAs [5]. This advantage is preserved for the Fast $(1+1)$ IA$_\gamma$ as shown by the following theorem.

Theorem 5. *The expected runtime of the Fast $(1+1)$ IA$_\gamma$ for Jump$_d$ and Cliff$_d$ is $O\left(\binom{n}{d}\cdot(d/\gamma)\cdot(1+\gamma\log n)\right)$.*

For Jump$_d$ and Cliff$_d$, the superiority of the Fast $(1+1)$ IA$_\gamma$ in comparison to the deterministic evaluations scheme (i.e., the original (1+1) IA) depends on the function parameter d. If $\gamma=\Omega(1/\log n)$, the Fast $(1+1)$ IA$_\gamma$ performs better when $\min\{d,n-d\}=o(n/\log n)$ while the deterministic scheme is preferable for larger $\min\{d,n-d\}$. However, for small $\min\{d,n-d\}$ the difference between the runtimes can be as large as a factor of n in favor of the Fast $(1+1)$ IA$_\gamma$, while even for the largest $\min\{d,n-d\}$, the difference is less than a factor of $\log n$ in favor of the deterministic scheme. Here we should also note that when both d and $n-d$ are in the order of $\Omega(n/\log n)$, 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. For these reasons the Fast $(1+1)$ IA$_\gamma$ should be more efficient in practice.

[2]In this paper we consider events to occur "with overwhelming probability" (w.o.p.) meaning that they occur with probability at least $1-2^{-\Omega(n)}$.

V. Fast Opt-IA$_\gamma$

In the previous sections we showed how the Fast $(1+1)$ IA$_\gamma$ achieves linear speed-ups in the exploitation phases compared to the traditional static HMP, while still maintaining a high quality performance at escaping from local optima of multimodal functions. In this section we will show how also the complete Fast Opt-IA$_\gamma$, which uses a population, cloning, hypermutations and an ageing operator, can take considerable advantage from the use of the *fast* HMP operator. In particular, we show linear, quasi-linear and exponential speed-ups compared to bounds on the expected runtime of the standard Opt-IA known in the literature.

A. Optimal Expected Runtimes for Unimodal functions

We start by analysing the performance of the Fast Opt-IA$_\gamma$ for standard unimodal benchmark functions, i.e., OneMax and LeadingOnes. Essentially the bounds derived previously for the Fast $(1+1)$ IA$_\gamma$ also apply to the Fast Opt-IA$_\gamma$ by multiplying them with the population and clone sizes as long as the parameter τ is set large enough such that ageing does not trigger with overwhelming probability before the global optimum is identified (i.e., the use of ageing does not make sense unless local optima are identified first). Hence, for correctly chosen parameter values, the algorithm can optimise these unimodal functions in optimal asymptotic expected runtimes.

Theorem 6. *Fast Opt-IA$_\gamma$ with parameters $\mu \geq 1$, $\lambda \geq 1$ and $\tau = c \cdot n \log n$ for some constant c, optimises* OneMax *and* LeadingOnes *in expected $O(\mu \cdot \lambda \cdot n \log n \cdot (1 + \gamma \log n))$ and $O(\mu \cdot \lambda \cdot n^2 \cdot (1 + \gamma \log n))$ fitness function evaluations respectively.*

B. Quasi-linear Speed-Ups when Both Hypermutations and Ageing are Necessary: HiddenPath

In [5], a benchmark function called HiddenPath (Fig. 5) was presented where the use of both the ageing and the hypermutation operators is crucial for finding the optimum in polynomial time. HiddenPath is defined as

$$\textsc{HiddenPath}(x) =$$

$$\begin{cases} n - \epsilon + \frac{\sum_{i=n-4}^{n} (1-x_i)}{n} & \text{if } |x|_0 = 5 \;\&\; x \neq 1^{n-5}0^5, \\ 0 & \text{if } |x|_0 < 5 \text{ or } |x|_0 = n, \\ n - \epsilon + \epsilon k/\log n & \text{if } 5 \leq k \leq \log n + 1 \;\&\; x = 1^{n-k}0^k, \\ n & \text{if } |x|_0 = n-1, \\ |x|_0 & \text{otherwise,} \end{cases}$$

where $|x|_0$ and $|x|_1$ respectively denote the number of 0-bits and 1-bits in a bit-string x. This function provides a gradient (where the fitness is evaluated by ZeroMax$= \sum_{i=1}^{n}(1-x_i)$) to local optima (i.e., solutions with $n-1$ 0-bits), from which the hypermutation operator can find another gradient (solutions with exactly five 0-bits with fitness increasing with more 0-bits in the rightmost five bit positions). This second gradient leads to a path which consists of $\log n - 3$ solutions of the form $1^{n-k}0^k$ for $5 \leq k \leq \log n + 1$ and ends up on the global optimum. This path (called Sp) is situated on the opposite side of the search space (i.e., nearby the complementary bit-strings of the local optima) so it can easily be reached with hypermutations. However, the ageing operator is necessary for the algorithm to accept a worsening; otherwise Sp is not accessible because the second gradient and the Sp path have lower fitness than that of the local optima.

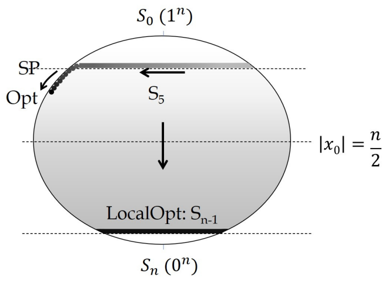

Fig. 3: HiddenPath [5]

In [5], an upper bound of $O(\tau\mu n + \mu n^{7/2})$ for the expected runtime of the traditional Opt-IA for the problem was established. The same proof strategy allows us to show an upper bound smaller by an $n/\log n$ factor for the Fast Opt-IA$_\gamma$. The smaller bound is achieved thanks to the speed-up that the *fast* HMP operator has in the exploitation phases. The speed-up is only quasi-linear rather than linear because of the $\gamma/2 = 1/(2\log n)$ probability of evaluating a successful 2-bit flip on the S_5 gradient leading towards the hidden path (i.e., hence the extra $O(\log n)$ term in the upper bound).

Theorem 7. *Fast Opt-IA$_\gamma$ requires $O(\tau\mu + \mu n^{5/2} \log n)$ fitness function evaluations in expectation to optimise* HiddenPath *with $\mu = O(\log n)$, $\lambda = 1$, $\gamma = \Omega(1/\log n) \leq 1/(5 \ln n)$ and $\tau = \Omega(n(\log n)^3)$.*

C. Exponential Speed-Ups when Traditional Hypermutations are Detrimental: Cliff$_d$

HiddenPath was especially designed to exploit the fact that HMP operators only stop at the first constructive mutation, hence always return the complementary bit-string with probability 1 unless some improvement over the parent is found before. On the other hand, by not returning solutions of lower quality apart from the complementary bit-string, the static HMP does not allow Opt-IA to take advantage of the power of ageing at escaping local optima in general, thus seriously limiting the potential explorative power of the algorithm. In this subsection we show that the Fast Opt-IA$_\gamma$, with appropriate parameter values for γ can escape from local optima by accepting a variety of solutions of lower quality.

For this purpose, we consider the Cliff$_d$ benchmark function (defined in the previous section) which is traditionally used to evaluate the performance of randomised search heuristics at escaping local optima by accepting solutions of lower quality [21]–[23]. Cliff$_d$ was also used to show the power of the ageing operator in [5]. RLS and EAs using standard bit mutation coupled with ageing can escape the local optimum of Cliff$_d$ by using their small mutation rates to create solutions at the bottom of the cliff in the same iteration where the rest of the population dies. This allows both algorithms to optimise the hardest CLIFF$_d$ functions (when the gap between the local and global optimum is linear, i.e., $d = \Theta(n)$) respectively in expected runtimes of $O(n \log n)$ and $O(n^{1+\epsilon} \log n)$ for any arbitrarily small positive constant ϵ. On the other hand, since static HMP with FCM does not return solutions of lower quality except for the complementary bit-string, the standard Opt-IA can only rely on hypermutations alone to escape from the local optima. Hence, the runtime is exponential in the distance between the top of the cliff and the global optimum w.o.p. The following theorem shows how for the hardest CLIFF$_d$ instances, i.e., $d = \Theta(n)$, the Fast Opt-IA$_\gamma$ has the best possible asymptotic expected runtime achievable by unary unbiased randomised search heuristics for any function with unique global optimum.

Theorem 8. *Fast Opt-IA$_\gamma$ with $\mu = O(\log n)$, $\lambda = O(\log n)$, $\gamma = 1/(n \log^2 n)$ and $\tau = \Theta(n \log n)$ needs $O(\mu \cdot \lambda \cdot \tau \cdot \frac{n^2}{d^2} + n \log n)$ fitness function evaluations in expectation to optimise* CLIFF *with $d \leq n/4 - \epsilon$ for a small constant ϵ.*

Note that the above result requires the parameter γ to be in the order of $\Theta(1/(n \log^2 n))$, while Lemma 1 implies that any $\gamma = \omega(1/\log n)$ suffices to keep the expected number of fitness function evaluations per hypermutation in an asymptotic order of at most $\Theta(1)$ (i.e., the algorithm does not waste more than a constant number of evaluations in each hypermutation). Nevertheless, the smaller $\gamma = 1/(n \log^2 n)$ is necessary for the algorithm to escape from the local optima efficiently. In particular, it allows the algorithm to only evaluate the first and/or the last bit flip until the optimum is found with high enough probability. This in turn allows the Fast Opt-IAγ to climb up the second slope before jumping back to the local optima via larger mutations. 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 9. *At least $2^{\Omega(n)}$ fitness function evaluations in expectation are executed before the Fast Opt-IA$_\gamma$ with $\gamma = \Omega(1/\log n)$ finds the optimum of* CLIFF$_d$ *for $d = (1-c)n/4$, where c is a constant $0 < c < 1$, independent of the values of μ, λ and τ.*

While the low parameter value allows the algorithm to escape from local optima as proven in Theorem 8, with such γ-values the hypermutation is in essence switched off, i.e., with high probability the algorithm only evaluates the first bit flip and the last one, with the latter being unlikely to be useful very often. We will address the problem again in Section VII, when discussing the best possible fitness evaluation distribution for the *fast* HMP operator for general purpose optimisation.

VI. FAST HYPERMUTATIONS FOR COMBINATORIAL OPTIMISATION

In the previous sections we used standard benchmark functions from the literature to show the speed-ups that can be achieved in the exploitation phases with the *fast* HMP operator while still maintaining excellent exploration capabilities at escaping local optima. In this section we will validate the gained insights using classical problems from combinatorial optimisation for which the performance of the traditional EAs and AISs is known in the literature.

In the following subsection we analyse the performance of the Fast $(1+1)$ IA$_\gamma$ for the NP-Hard PARTITION problem. Static HMP operators allow AISs to efficiently find arbitrarily good constant approximations for the problem [9]. This is achieved by escaping local optima of low quality by flipping approximately half of the bits. Given that the parabolic distribution of the *fast* HMP operator decreases the probability of evaluating solutions as the $n/2$th bit flip is approached, it wouldn't be surprising if the Fast $(1+1)$ IA$_\gamma$ was to struggle on this problem. Nevertheless, we will present the remarkable result that a linear factor smaller upper bound on the expected runtime can be achieved by the algorithm compared to the static HMP even in this apparently unfavourable scenario. This result shows that the insights gained from the analysis of HIDDENPATH, that speed-ups may be achieved for multimodal problems through faster exploitation, may also appear in NP-Hard problems with numerous real-world applications.

In Section VI-B, we turn to the VERTEX COVER problem. We will rigorously prove linear speed-ups of the Fast $(1+1)$ IA$_\gamma$ to identify feasible solutions to the problem compared to static HMP using a node-based representation, and for identifying 2-approximations for any instance of the problem using an edge-based representation. Thus, the analysis confirms the greater exploitative capabilities of the Fast HMP operators.

At the end of each subsection we will also argue how the results also hold for the population-based Fast Opt-IA$_\gamma$.

A. PARTITION

PARTITION, or NUMBER PARTITIONING, is a simple makespan scheduling problem where the goal is to schedule n different jobs with processing times $p_1 \geq p_2 \geq \cdots \geq p_n$ on two identical machines in a way that the load of the fullest machine is minimized. It is considered to be one of the six basic NP-complete problems [24] which arises in many real world applications such as allocation tasks in industry and in information processing systems [25], [26]. It is known that the $(1+1)$ EA and RLS get stuck on approximately 4/3 approximation ratios on worst-case instances of the problem. However, they can find a $(1+\epsilon)$ approximation for any $\epsilon = \Theta(1)$ if an appropriate restart strategy that depends on the chosen ϵ is put in place [27]. On the other hand the (1+1) IA, by using static HMP can escape the local optima where EAs and RLS get stuck, thus solving the worst-case instance for EAs in expected $O(n^2)$ time. As a result it finds arbitrarily good approximations with an expected runtime that is only exponential in ϵ, i.e., it can efficiently identify arbitrarily small constant $(1+\epsilon)$ approximations in every run in expected time $O(n^3)$ [9]. In the following two subsections we use the same proof techniques used in [9] to prove that the Fast $(1+1)$ IA$_\gamma$ optimises the worst-case instance for EAs in expected time $O(n \log n)$ and identifies a $(1+\epsilon)$ approximation in expected time $O(n^2)$, thus providing upper bounds that are respectively a quasi-linear and linear factors smaller than those derived for the traditional static HMP operator.

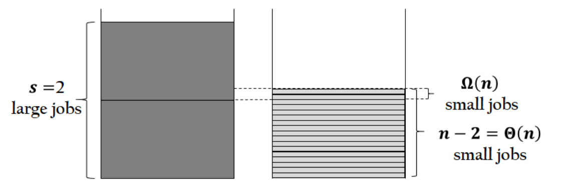

Fig. 4: Worst-case approximation PARTITION instance, W_ϵ, for EAs [27].

1) EA's Worst-Case Instance Class: The worst-case instance W_ϵ for the $(1+1)$ EA is depicted in Figure 4. It consists of two large jobs p_1 and p_2 each with processing times $(1/3-\epsilon/4)$, and $n-2$ small jobs, $p_3, p_4, \ldots, p_n$, each with a processing time of $(1/3+\epsilon/2)/(n-2)$. The total processing time is normalised between 0 and 1, and the global optima, consisting of one large job and half of the small jobs on each machine, have a makespan of $1/2$. It has been shown that with constant probability the $(1+1)$ EA and RLS take $n^{\Omega(n)}$ fitness function evaluations to find a solution better than $(4/3-\epsilon)$ approximation for W_ϵ [27].

The Fast $(1+1)$ IA using static HMP has been proved to be able to efficiently optimise the instance in $O(n^2)$ expected runtime [9]. The following theorem shows how the Fast $(1+1)$ IA$_\gamma$ can optimise it in $O(n \log n)$ expected function evaluations if it uses any parameter value $\gamma = \Omega(1/\log n)$. The speed-up is simply due to the fewer function evaluations wasted in the exploitation phases (i.e., it hillclimbs up to the local optima in $O(n \log n)$ expected evaluations rather than $O(n^2)$. While it is a logarithmic factor slower at escaping from the local optima, this burden does not increase the overall asymptotic order.

Theorem 10. *The Fast $(1+1)$ IA$_\gamma$ optimises W_ϵ in $O(\frac{n}{\gamma} + n\log n)$ expected fitness function evaluations.*

2) Worst-Case Approximation Ratio: We now prove the main result of this subsection.

Theorem 11. *The Fast $(1+1)$ IA$_\gamma$ finds a $(1+\epsilon)$ approximation for any instance of* PARTITION *in* $\left[2en^2 \cdot (2^{2/\epsilon}+1) + (\epsilon^{(2/\epsilon)+1})^{-1}(1-\epsilon)^{-2}e^3 2^{2/\epsilon}\frac{n}{2\gamma}\right]$ $\cdot(1+\gamma\log n)$ *expected fitness function evaluations for any $\epsilon = \omega(1/\sqrt{n})$.*

For $\gamma = 1/\log n$, as recommended herein for the Fast $(1+1)$ IA$_\gamma$, the expected runtime is dominated by the term $2en^2 2^{2/\epsilon}$. Hence the upper bound is a linear factor smaller than that of the (1+1) IA using traditional static HMP. We remark that even though the Fast $(1+1)$ IA$_\gamma$ is a logarithmic factor slower at escaping from the local optima, a speed-up is still achieved because the dominating term is due to the expected time to hill-climb up to the local optima, a task at which the FCM$_\gamma$ operator is considerably faster. Hence, this advantage dominates even in the PARTITION scenario where flipping approximately $n/2$ bits is essential to escape local optima via mutation and detrimental to the Fast $(1+1)$ IA$_\gamma$.

We point out that the complete Fast Opt-IA$_\gamma$ can also solve the worst-case instance to optimality and identify the approximation ratios by either using the ageing operator to restart the search process when trapped on local optima (with optimisation time $O(n^2)$ [9]) or by escaping them via hypermutation. Hence, the Fast Opt-IA$_\gamma$ can take advantage of both hypermutations *and* ageing to efficiently overcome the local optima of PARTITION.

B. VERTEX COVER

In this section we will use the NP-Hard VERTEX COVER problem to rigorously prove that the Fast $(1+1)$ IA$_\gamma$ can take advantage of FCM$_\gamma$ to achieve considerable speed-ups compared to static HMP on another classic problem from combinatorial optimisation with numerous real-world applications [28] in, e.g., classification methods [29], computational biology [30], and electrical engineering [31], [32].

Given an undirected graph $G = (V, E)$, the VERTEX COVER problem asks to find a minimal subset of vertices, $V' \subseteq V$, such that every edge $e \in E$ is adjacent to one of the vertices in V'. Any set of vertices such that all edges in the graph are adjacent to at least one vertex in the set is a feasible solution and is called a *cover*. The aim of the problem is to identify the cover of minimal size (i.e., the minimum vertex cover). While the problem is NP-Hard, hence no algorithm is expected to be efficient for every instance, we will show that the (1+1) IA using the traditional HMP operator is particularly slow at identifying any cover and how the Fast HMP operators speed-up the algorithm by a linear factor when node-based representations are used. In the next subsection, we will prove the same linear speed-up for identifying 2-approximations when edge-based representations are employed.

1) Node-Based Representation: We will use the commonly applied fitness function over node-based representations [33]–[35]. Candidate solutions are bit-strings of length $|V| = n$, where each bit x_i is associated to a node in the graph and is set to 1 if the vertex i is included in the cover set, and to 0 otherwise. The fitness of a candidate solution is,

$$f_v(x) = \sum_{i=1}^{n}\left(x_i + n(1-x_i)\sum_{j=1}^{n}(1-x_j)e_{i,j}\right),$$

where $e_{i,j}$ takes value 1 if there is an edge connection between vertex i and vertex j in the graph G. This fitness function sums the number of vertices in the cover (the first term) and gives a large penalty to the number of uncovered edges (the second term).

It is well-known that both the $(1+1)$ EA and RLS can find feasible covers in expected time $\Theta(n \log n)$. The following theorem shows that the $(1+1)$ IA using the traditional static HMP operator is a linear factor slower.

Theorem 12. *The expected time until the $(1+1)$ IA finds a vertex cover using the node-based representation and f_v is $\Theta(n^2 \log n)$.*

We now prove that the Fast $(1+1)$ IA$_\gamma$ is a linear factor faster.

Theorem 13. *The expected time until the Fast $(1+1)$ IA$_\gamma$ finds a vertex cover using the node-based representation and f_v is $\Theta(n \log n \cdot (1+\gamma\log n))$.*

2) Edge-Based Representation: It is well understood that using the node-based representation of the previous subsection, RLS and EAs may get stuck on arbitrarily bad approximations for the VERTEX COVER problem [34], [35]. In [36], it was shown that 2-approximations may be guaranteed by these algorithms if an edge-based representation is employed, such that if an edge is selected, then both its endpoints are included in the cover. For the approximation to be guaranteed, it is necessary to give a large penalty to adjacent edges, i.e., the fitness decreases considerably if adjacent edges are deselected. Given a graph $G = (V, E)$ with $|V| = n$ and $|E| = m$ and an edge-based representation where solutions are bit-strings of length m, the fitness function is,

$$\begin{aligned} f_e(x) = f_v(x) + (|V| + 1) \cdot (m + 1) \\ \cdot |\{(e, e') \in E(x) \times E(x) | e \neq e', e \cap e' \neq \emptyset\}|. \end{aligned}$$

We will now prove that while with this representation the $(1+1)$ IA with traditional static HMP requires super-quadratic expected runtime in the number of edges to find a 2-approximation in the worst-case, the Fast $(1+1)$ IA$_\gamma$ guarantees 2-approximations in expected time $O(m \log m)$ for any instance of the problem.

Theorem 14. *Using the edge-based representation and fitness function f_e, the $(1+1)$ IA has an expected runtime of $\Omega(m^2 \log m)$ to find a 2-approximation for vertex cover. The Fast $(1+1)$ IA$_\gamma$ finds a 2-approximation within $O((m \log m) \cdot (1 + \gamma \log m))$ expected fitness function evaluations.*

As long as the ageing parameter τ is set to be asymptotically larger than the expected waiting time of the improvement with smallest probability, all the VERTEX COVER results can easily been shown to also hold for the Fast Opt-IA$_\gamma$ by multiplying the upper bounds with the population and clone sizes.

VII. OPTIMAL PROBABILITY DISTRIBUTIONS

In the following subsection we compare the advantages and disadvantages of our proposed *fast* HMP operators to other '*fast* mutation' operators from the literature. In the subsequent subsection we draw on the gained insights to provide the best parameter settings for the *fast* HMP operators for black box scenarios where limited problem knowledge is available.

A. Comparison with Fast Evolutionary Algorithms

While high mutation rates are typical of an immune system response, they do not occur naturally in Darwinian evolutionary processes. Indeed, low mutation rates are essential in traditional generational evolutionary and genetic algorithms to avoid exponential runtimes on any function of practical interest [37]. However, increasing evidence is mounting that higher mutation rates than standard are beneficial to steady-state GAs both for exploitation (i.e., hillclimbing) [38], [39] and exploration (i.e., escaping from local optima) [40]. These high mutation rates have been made possible by taking advantage of the artificially introduced elitism in steady-state EAs [41]. Furthermore, it has been recently shown how the selective pressure can be more accurately controlled in steady-state EAs, and decreased below what is possible in generational models, in turn allowing to achieve a better exploration/exploitation balance in evolutionary search [42]. Such insights have recently been exploited in the evolutionary computation community in the design of so-called *fast* EAs that use *heavy tailed* mutation operators to allow a larger number of bit flips more often than the standard bit mutations (SBM) traditionally used in EAs. By using higher mutation rates, *fast* EAs can provably escape from local optima more efficiently than the traditional SBM. Since these analogies are very similar to the insights gained in this paper, and in AISs previous works, in this section we compare the performance of the *fast* HMP operator to those of the *fast* EAs.

Two heavy tailed mutation-based EAs for discrete optimisation have been recently introduced. In the first one, which we call Fast (1+1) EA$_\beta$, the tail of the probability distribution follows a power law [14] (i.e., the probability that larger number of bits flip decreases slower than in SBM). In the second one, which we call Fast (1+1) EA$_{\text{UNIF}}$, the tail is uniformly distributed [16]. To illustrate their advantages over SBM at escaping from local optima, these works have naturally used the JUMP$_d$ function, just like traditionally for AISs. Thus, we will start by comparing their performance versus that of the Fast $(1+1)$ IA$_\gamma$ for JUMP$_d$. We begin with the latter algorithm, as the analysis of the former will motivate the optimal settings for the hypermutation distribution that we will present in the next subsection for typical black box scenarios where minimal problem knowledge is assumed.

1) Uniform Heavy Tailed Mutations [16]: The Fast (1+1) EA$_{\text{UNIF}}$ uses the following distribution:

$$p_i = \begin{cases} p & \text{for } i = 1, \\ (1-p)/(n-1) & \text{for } 1 < i \leq n. \end{cases} \tag{2}$$

where p_i is the probability that i bits flip and $p = \Theta(1)$ is a constant, e.g., $1/e$.

This operator has a very similar behaviour to the original static HMP operator with FCM since over n fitness function evaluations both operators evaluate the same expected number of solutions at Hamming distance k (for any $k \neq 1$) except for a factor of $(1 - p)$.

Just like the (1+1) IA and the Fast $(1+1)$ IA$_\gamma$, the Fast (1+1) EA$_{\text{UNIF}}$ can easily explore the opposite side of the search space and can even obtain polynomial expected runtimes if the jump size is in the order of $n - O(1)$. However, just like for the traditional HMP operator, the drawback of this approach is that it is slower than the Fast $(1+1)$ IA$_\gamma$ for jump sizes $d < n/\log n$ and $d > n - n/\log n$. The intuition is that the Fast (1+1) EA$_{\text{UNIF}}$ assigns a constant probability p only to 1-bit flips while assigning a probability in the order of $\Omega(1/n)$ to all others. Hence, similarly to the traditional static HMP operator, a solution at the correct distance d to the parent is only sampled once every n fitness function evaluations resulting in the same asymptotic performance for all possible $d > 2$. In particular, while for small and large d (where efficient performance is achievable), the detriment in performance is as large as a factor of n, for other values of d, the difference of performance is in favour of the Fast (1+1) EA$_{\text{UNIF}}$ by at most a factor of $\Theta(\log n)$. This, however, has no realistic influence on the applicability of the algorithm, since the expected runtime to perform such jumps is exponential in the problem size in any case. Hence, the Fast $(1+1)$ IA$_\gamma$ is superior at escaping local optima, while both algorithms display the same hillclimbing performance (i.e., they both flip and evaluate exactly one bit with constant probability $p = \Theta(1)$).

2) Power Law Heavy Tailed Mutations [14]: The (1+1) EA$_\beta$ uses a heavy tailed standard bit mutation operator (i.e., it flips each bit with probability χ/n). The mutation rate χ is sampled in each step with probability,

$$p(\chi) = \frac{\chi^{-\beta}}{\sum_{i=1}^{n/2} i^{-\beta}}$$

where the parameter β is assumed to be a constant strictly greater than 1 to ensure that the sum $C^{\beta}_{n/2} := \sum_{i=1}^{n/2} i^{-\beta}$ is in the order of $O(1)$.

The optimal expected runtime for SBM operators to optimise JUMP$_d$ is $\frac{n^n}{d^d(n-d)^{n-d}}$, which is achieved by using the optimal mutation rate d/n which can only be applied if the jump size d is known in advance. Naturally, in a black-box scenario this parameter of the problem is not known to the algorithm. The above mutation operator was explicitly designed to have an adequate compromised performance over all possible values of d.

The Fast (1+1) EA$_\beta$ has an expected runtime of $\Theta\left(d^\beta\binom{n}{d}\right)$ on the JUMP$_d$ function which differs from the best possible expected runtime by at most a factor of $\Theta(d^{\beta-0.5})$. The Fast $(1+1)$ IA$_\gamma$ evaluates a solution at Hamming distance d with probability γ/d in each hypermutation, and wastes the remaining $\Theta(\gamma \log n)$ expected evaluations, resulting in an expected waiting time of $\Theta(d\binom{n}{d}\log n)$. Thus, the Fast $(1+1)$ IA$_\gamma$ has an extra $\Theta(\log n)$ factor in its runtime for constant jump sizes. In particular, since the Fast (1+1) EA$_\beta$ uses a power law distribution, for any jump of size $d = \Theta(1)$, the probability that the operator picks the mutation rate d/n which gives the highest improvement probability is in the order of $d^{-\beta} = \Theta(1)$ when $d = \Theta(1)$.

However, the algorithm struggles with larger jump sizes compared to the Fast $(1+1)$ IA$_\gamma$. This is particularly critical for very large jumps i.e., $d = n - O(1)$, where the Fast $(1+1)$ IA$_\gamma$ has polynomial expected runtime $O(d\binom{n}{n-d}\log n)$ while the Fast (1+1) EA$_\beta$ has exponential runtime because it flips bits with probability at most $\chi/n = 1/2$ by design (as a larger mutation rate was deemed unnecessary in the original work). If the cap on the maximum mutation rate is removed (as was recently considered in [15]), the resulting operator can also achieve polynomial expected runtimes for extremely large jump sizes. However, due to the power law distribution, the probability of flipping $n - O(1)$ bits is in the order of $O(n^{-\beta})$ which results in a polynomially slower expected performance to that of the Fast $(1+1)$ IA$_\gamma$. This is due to the symmetric sampling distribution of the FCM$_\gamma$ operator around $n/2$ that allows considerably larger probabilities of evaluating offspring at distance $n - O(1)$.

Overall, the Fast $(1+1)$ IA$_\gamma$ is asymptotically faster at escaping from local optima for all super-logarithmic jump sizes and is at most a $\Theta(\log n)$ factor slower for small constant jumps. In the next subsection we will show how to reduce the logarithmic factor in the Fast $(1+1)$ IA$_\gamma$ to just a constant while maintaining its advantage in the settings where it has better performance.

Nevertheless, we now show that the Fast (1+1) EA$_\beta$ can still be very efficient in practice at escaping from local optima with large basins of attraction. In particular, just like the Fast $(1+1)$ IA$_\gamma$, it has an $O(n^2)$ expected runtime to find arbitrarily good constant approximations for the PARTITION problem considered in Section VI-A.

Theorem 15. *The Fast* $(1+1)$ *EA*$_\beta$ *finds a* $(1+\epsilon)$ *approximation for any* PARTITION *instance in* $2(C^{\beta}_{n/2})en^2 \cdot (2^{2/\epsilon}+1) + (C^{\beta}_{n/2})(n(\epsilon-\epsilon^2))^\beta \cdot \epsilon \cdot (\epsilon-\epsilon^2)^{-2/\epsilon}$ *expected fitness function evaluations (for any* $\epsilon = \omega(1/\sqrt{n})$*).*

Even though the Fast $(1+1)$ EA$_\beta$ is slower at jumping over large basins of attraction, its expected runtime for PARTITION is dominated by the expected time spent in the hillclimbing phases. Indeed the bounds on the expected runtimes during exploitation of the Fast $(1+1)$ EA$_\beta$ and the Fast $(1+1)$ IA$_\gamma$ are asymptotically the same (i.e., they differ in the former having an extra $C^{\beta}_{n/2} = \Theta(1)$ factor and the latter an extra factor of $O(1+\gamma\log n)$ which is $O(1)$ for $\gamma = 1/\log n$). Concerning the terms related to the expected times to escape from local optima, the Fast $(1+1)$ IA$_\gamma$ has an asymptotically smaller term of $2^{2/\epsilon} \cdot \frac{n}{\gamma}$ compared to the $(n(\epsilon-\epsilon^2))^\beta$ term for some constant $\beta > 1$ for the Fast $(1+1)$ EA$_\beta$. We should note here that the $2^{2/\epsilon}$ factor (i.e., exponential in $1/\epsilon$) may appear to possibly make a crucial difference for small constant approximations in practice. However, on one hand, this is likely to be overly pessimistic since it assumes that whenever the hypermutation is about to find an approximation, another improvement prevents it from flipping the necessary number of bits. On the other hand, the exponential factor nevertheless appears for both algorithms in the dominating term related to the hillclimbing phases.

We now highlight a huge advantage of the Fast $(1+1)$ EA$_\beta$ over the *fast* HMP operators when escaping local optima in conjunction with ageing by accepting solutions of lower quality. In Section V we proved that the Fast Opt-IA$_\gamma$ optimises the Cliff$_d$ function efficiently, if the parameter γ of the FCM$_\gamma$ is set to extremely small values in the order of $\gamma = \Theta(1/(n\log^2 n))$ (Theorem 8). As a result the algorithm very rarely evaluates solutions where more than one bit is flipped i.e., it essentially does not hypermutate anymore. The following theorem shows how the Fast $(1+1)$ EA$_\beta$ can optimise the function efficiently while still mutating many bits very often i.e., it hypermutates. The result comes at the expense of slightly increasing the power law parameter to a constant $\beta > 2$ and at the expense of a square root term in the upper bound of the expected runtime instead of the logarithmic term that appears in the expected runtime of the Fast Opt-IA$_\gamma$ with small γ. Nevertheless, although not optimal for JUMP$_d$, with such a parameter setting the algorithm is only a constant factor slower for the JUMP$_d$ instances for which it is very efficient (i.e., $d = \Theta(1)$).

Theorem 16. *The Fast* $(1+1)$ *EA*$_\beta$ *with hybrid ageing parameter* $\tau = \Omega(n\log n)$ *and* $\beta \geq 2+\epsilon$ *needs* $O(\tau \cdot n^{3/2})$

fitness function evaluations in expectation to optimise CLIFF *with any linear $d \leq n(1/4 - c)$ for any arbitrarily small positive constants ϵ and c.*

B. Power-Law Hypermutations

In the previous subsection we highlighted two advantages of the power law heavy tailed mutation operator of the Fast $(1+1)$ EA$_\beta$ over the *fast* HMP operator introduced in this paper. Firstly, the former operator jumps out of local optima with small basins of attraction faster by a logarithmic factor at the expense of being slower for larger basins of attraction. Secondly, it allows to escape local optima together with ageing by accepting solutions of lower fitness while still keeping quite high mutation rates i.e., the Fast Opt-IA$_\gamma$ has to reduce it to at most that of SBM. These advantages are due to the capability of the power-law distribution of balancing well the number of large and small mutations. In this subsection we will identify an "optimal" evaluation distribution for the *fast* HMP operator such that it can take advantage from the balancing capabilities of the power-law distribution while keeping its own advantages when larger basins of attraction have to be overcome.

In particular, considering the power-law distribution's poor performance for JUMP$_d$ functions with gap sizes of $d = \Omega(\log^{\frac{1}{\beta-1}} n)$, and especially for $d = n(1 - o(1))$, we keep the symmetry of the *fast* HMP operator around $n/2$ bit flips, but increase and decrease the evaluation probabilities away and to $n/2$ following a power-law. Just like in the Opt-IA literature, we will present variants with and without FCM and call the power-law HMPs FCM$_\beta$ and HMP$_\beta$, and the resulting algorithms Fast (1+1) IA$_\beta$ and Fast Opt IA$_\beta$ respectively, according to whether they use populations and ageing or not (we will see that the performance of FCM$_\beta$ and that of HMP$_\beta$ are approximately equivalent so we intentionally do not state whether the Fast (1+1) IA$_\beta$ uses one operator or the other as it does not affect the results we present i.e., either can be used).

Recall that the parameter β of the Fast $(1+1)$ EA$_\beta$ is assumed to be a constant strictly greater than 1 to ensure that the sum $\sum_{i=1}^{n/2} i^{-\beta}$ is in the order of $O(1)$. Thus, any particular mutation rate χ has a probability of being picked in the order of $\Theta(\chi^{-\beta})$. Notice that if we were to set the parameter to $\beta = 1$, the power-law mutation operator would have a very similar behaviour to that of the Fast $(1+1)$ IA$_\gamma$. In particular, the resulting operator would pick a mutation rate χ with probability $1/(\chi \ln n)$.

Similarly, FCM$_\gamma$ with $\gamma = 1$ evaluates a solution with Hamming distance $k \neq 1$ away from the parent with probability $1/k$ and every call of the operator evaluates roughly $\ln n$ solutions in expectation. Thus, when compared over $\Theta(\log n)$ consecutive fitness function evaluations, the expected number of offspring k bits away from their parent are in the same asymptotic order. However, the parameter γ of the Fast $(1+1)$ IA$_\gamma$ scales the frequency of evaluations at Hamming distance k by the same multiplicative factor for all k, while the parameter β of the Fast $(1+1)$ EA$_\beta$ controls the emphasis on the smaller mutations. In particular, for $k \in \{2, \ldots, \frac{n}{2} - 1\}$ changing β changes the conditional probability of flipping k bits given that either k or $k{+}1$ bits are flipped, while changing γ still conserves the ratio of sampled solutions with distance k and $k+1$.

These considerations lead us to believe that the ideal symmetric distribution for the HMP operator is a power-law one, where we move the probability mass further towards $\omega(1)$ bit flips, compared to the Fast $(1+1)$ EA$_\beta$:

$$p_i := \frac{(\min\{i+1, n-i+1\})^{-\beta}}{\sum_{k=0}^{n}(\min\{k+1, n-k+1\})^{-\beta}}.$$

Here the parameter should be set such that $\beta \geq 1$. We denote the denominator of the above expression as $H_n^\beta := \sum_{k=0}^{n}(\min\{k+1, n-k+1\})^{-\beta}$

With $\beta = 1$, the probability distribution for $i > 1$ is identical to that of FCM$_\gamma$ for the parameter value we have used throughout the paper i.e., $\gamma = 1/\log n$. Notice that if $\beta = 0$ was allowed, then the probability that i bits flip would be uniformly distributed at random i.e., the operator would become very similar to that of the (1+1) EA$_{\text{UNIF}}$ which flips i bits with probability $\Theta(1)/(n-1)$ for each $i > 1$. We have discussed why this is an inconvenient distribution in the previous section. Note that the original heavy-tailed mutation operator first picks the mutation rate with which each bit position is flipped independently. Since we directly pick the number of bit-positions to be flipped, we assign a positive probability to not flipping any bits. This allows the operator to copy the best individuals and plays a critical role in the performance of population-based algorithms [37]–[40], [43].

The operator behaviours, with and without FCM, are similar but not identical. While the HMP$_\beta$ operator evaluates exactly one new offspring per operation, the number of evaluated solutions per hypermutation of the FCM variant, FCM$_\beta$, is randomly distributed with expectation 1 (i.e., more than one evaluation – or zero – may occur in one hypermutation: the behaviour is exactly the same as in Definition 1 but using the power law distribution). A comparison between the power-law distributions of the mutation operators of the (1+1) EA$_\beta$, the symmetric ones of the (1+1) IA$_\beta$, the (1+1) EA$_{\text{UNIF}}$ and the traditional SBM are shown in Figure 5. Note that for the (1+1) EA$_\beta$ we have extended the probability distribution range from [14] to n and considered the variant which flips exactly $k \in \{1, \cdots, n\}$ bits after the mutation size is determined (similarly to what has been considered in [15]) rather than independently flipping all bit positions with probability k/n.

Figure 6 shows a comparison of the expected runtimes of the (1+1) IA$_\beta$ and the (1+1) EA$_\beta$ to escape from local optima with different basins of attraction. Without loss of generality we assume that the local optimum is located at the 0^n bit-string (i.e., the red dot). Let us denote with $y \in \{0,1\}^n$ the unique global optimum which has a higher fitness value than x and $k := HD(x, y)$. The black dots represent different potential positions in the search space for the global optimum. The circles around the potential global optima represent basins of attraction which may or may not have higher fitness than the local optimum. These are nevertheless reachable via ageing by accepting lower quality solutions (as we have shown for HIDDENPATH and CLIFF).

Regardless of the mutation operator employed by the algorithm, the probability that x is mutated into y is at most

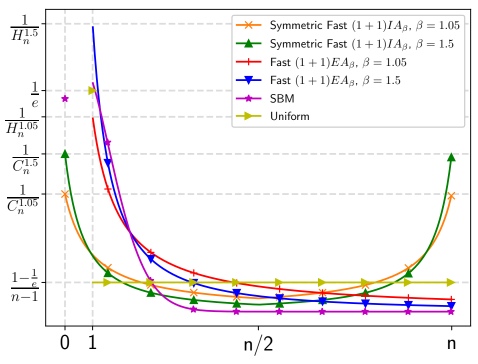

Fig. 5: The probability of flipping exactly k bits for the extended heavy-tailed mutation operator of Fast (1+1) EA$_\beta$ (red and blue) and the symmetric heavy tailed mutation operator of Fast (1+1) IA$_\beta$ (green and orange) for different β values. The SBM used by standard EAs (purple) and the uniform heavy tailed mutation of Fast (1+1) EA$_{\text{UNIF}}$ [16] with $p = 1/e$ (yellow) are added for comparison. The input size is set to $n = 14$ for visualisation.

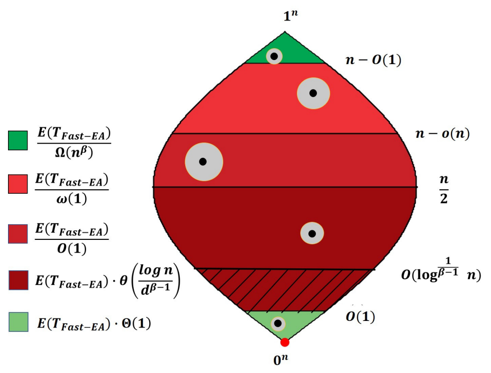
1^n
$n - O(1)$
$n - o(n)$
$\frac{n}{2}$
$O(\log^{\frac{1}{\beta-1}} n)$
$O(1)$
0^n
🟩 $\frac{E(T_{Fast-EA})}{\Omega(n^\beta)}$
🟥 $\frac{E(T_{Fast-EA})}{\omega(1)}$
🟥 $\frac{E(T_{Fast-EA})}{O(1)}$
🟥 $E(T_{Fast-EA}) \cdot \theta\left(\frac{\log n}{d^{\beta-1}}\right)$
🟩 $E(T_{Fast-EA}) \cdot \Theta(1)$

Fig. 6: A description of the performance comparison of the Fast (1+1) IA$_\beta$ and the Fast (1+1) EA$_\beta$ at escaping from a local optimum placed on the hypercube at 0^n w.l.o.g. The global optima (and basins of attraction of any fitness quality) are located in example positions. For both algorithms the same $\beta > 1$ holds for all regions except for the darkest red area. For the latter area, the Fast (1+1) IA$_\beta$ uses the best possible parameter value $\beta = 1$. For equal $\beta > 1$, the Fast (1+1) IA$_\beta$ would be a constant factor slower than the Fast (1+1) EA$_\beta$. For both parameter setting cases, the Fast (1+1) EA$_\beta$ asymptotically outperforms the Fast (1+1) IA$_\beta$ in the hatched area only.

$\binom{n}{k}^{-1}$ since for an unbiased mutation operator all individuals with distance k to the parent have an equal probability to be sampled and $\binom{n}{k}$ is the number of individuals with Hamming distance k to x. Note here that the binomial coefficients satisfy $\binom{n}{k} = \binom{n}{n-k}$ for all $k \leq n$. Thus, if both k and $n - k$ are in the order of $\omega(1)$, the mutation probability is superpolynomially small and the jump from x to y has a superpolynomial expected time (i.e., the shades of red areas in the figure). Even if we relax our scenario such that the solution y has a basin of attraction of a constant size, *i.e.*, all individuals $z \in \{0,1\}^n$ with $HD(y,z) < d$ for some constant d lead to y by hillclimbing, the expected time to escape the local optima would still be super-polynomially large. For this reason we modify the distribution over $\{0, 1, \dots, n\}$ used to determine how many bits the heavy-tailed mutation operator will flip. We shift the probability mass from the middle to the extremities (*i.e.*, from around $n/2$ to near 0 and n): away from mutation sizes where a polynomial expected time is not possible.

Overall, for any $k = \Theta(1)$ the heavy-tailed mutation operator in [14] is only faster by a constant factor than the newly suggested power-law symmetric operators at escaping the local optimum. Only for $k = O(\log^{\frac{1}{\beta-1}} n)$ (i.e., the hatched area in the figure), it is slightly asymptotically faster where both operators have super-polynomial expected runtime. On the other hand, for all other distances of the basin of attraction of the global optimum, the symmetric power-law mutation operator is faster. In particular, the heavy-tailed operator is a polynomial factor slower than the symmetric one when $n - k$ is in the order of $o(n)$, including for $n - k = O(1)$ where the expected runtimes of the operators are polynomial. Hence, for ranges of k where a polynomial expected waiting time is possible, the heavy-tailed operator of the (1+1) EA$_\beta$ is either faster by only a constant factor than the symmetric one (i.e., when k is constant) or slower by a polynomial factor (i.e., when $n - k$ is a constant). We point out that if in the "super-polynomial space" (i.e., the red areas in the figure) the basins of attraction were large enough to allow for polynomial expected waiting times, then the Fast (1+1) IA$_\beta$ would still be faster than the Fast $(1 + 1)$ EA$_\beta$ except for basins that fall into the hatched area.

Compared to the Fast $(1 + 1)$ IA$_\gamma$, the Fast (1+1) IA$_\beta$ is faster for all jump sizes for appropriate parameter settings (i.e., $\beta = 1$ for $k < \log^{\frac{1}{\beta-1}} n$ and $\beta > 1$ otherwise) at the expense of being a constant factor slower at hillclimbing for the suggested values of β (i.e., close to $\beta = 1$). In particular, the (1+1) IA$_\beta$ is a logarithmic factor faster than the Fast $(1+1)$ IA$_\gamma$ for jumps in the "polynomial space" (i.e., the green areas in the figure).

Naturally, the described above scenario also includes the behaviour on the JUMP function. The behaviour of the FCM$_\beta$ operator for escaping local optima combined with ageing, by accepting solutions of inferior fitness, requires a more precise analysis. Theorem 16 regarding the (1+1) EA$_\beta$ with ageing for CLIFF$_d$ relies on the distribution over the mutation rate to monotonically decrease. Since the distribution of the symmetric operator starts increasing for mutation sizes larger than $n/2$, the result does not transfer directly the (1+1) IA$_\beta$. In particular, large mutation rates may lead the algorithm to jump back to the local optima once it has escaped. Since the previous results hold for gap sizes $d \leq (1 - c)n/4$ for any constant c, we will show that bit flips in the order of $n(1 - o(1))$ only produce solutions with smaller fitness than those observed on the second slope of the function, *i.e.*, solutions with more than $n - d$ 1-bits. Hence the operator is efficient for the function class coupled with ageing. The following theorem shows that FCM$_\beta$ (or HMP$_\beta$) are better suited than FCM$_\gamma$ to be used in the complete Opt-IA since they have high mutation rates (i.e., they hypermutate) and work well in harmony with ageing, as originally desired in the design of the Opt-IA.

Theorem 17. *The Fast (1+1) IA$_\beta$ with hybrid ageing parameter $\tau = \Omega(n \log n)$ and $\beta \geq 2 + \epsilon$ needs $O(\tau \cdot n^{3/2})$ fitness function evaluations in expectation to optimise* CLIFF$_d$ *with any linear $d \leq n(1/4 - c)$ for any arbitrarily small positive constants c and ϵ.*

The performance of the (1+1) IA$_\beta$ on the other functions analysed in the previous sections is straightforward to bound. For the PARTITION problem the expected runtime differs by at most a constant factor from that of the Fast $(1+1)$ EA$_\beta$ if both algorithms use the same β parameter.

Theorem 18. *Let $H_n^\beta := \sum_{i=0}^{n} (\min(i+1, n-i+1))^{-\beta}$. Then, the Fast (1+1) IA$_\beta$ finds a $(1+\epsilon)$ approximation for any* PARTITION *instance in $2(H_n^\beta)en^2 \cdot (2^{2/\epsilon}+1) + (H_n^\beta)^{-1}(n(\epsilon - \epsilon^2))^\beta \cdot \epsilon \cdot (\epsilon - \epsilon^2)^{-2/\epsilon}$ expected fitness function evaluations. (for any $\epsilon = \omega(1/\sqrt{n})$).*

Thus, the expected runtime of the (1+1) IA$_\beta$ is in the order of $O(n^2)$ for $1 < \beta \leq 2$. For ONEMAX and LEADINGONES, its expected runtime asymptotically matches the best possible achievable by unbiased unary randomised search heuristics due to the constant probability of flipping a single bit for any constant $\beta > 1$. If coupled with ageing, a logarithmic factor may be shaved off from the upper bound on the expected runtime of the (1+1) IA$_\beta$ for the HIDDENPATH function compared to that of the Fast Opt-IA$_\gamma$. This is due to the higher probability of the (1+1) IA$_\beta$ of performing 2-bit flips on the slope leading to the hidden path. The only advantage of FCM$_\gamma$ over the symmetric power law operator appears for the CLIFF$_d$ function which the former can optimise in expected $O(n \log n)$ fitness evaluations if is used with ageing, while we could only bound the expected runtime for the (1+1) IA$_\beta$ by $O(\tau \cdot n^{3/2})$. Recall that for the $O(n \log n)$ bound, a very small γ value is required, effectively reducing the hypermutation operator FCM$_\gamma$ to perform single bit flips most of the time. A similar behaviour may be achieved by the (1+1) IA$_\beta$ by increasing its parameter value to $\beta = \Omega(\log n)$. However the same drawbacks as for the (1+1) IA$_\gamma$ would be obtained i.e., the algorithm would rarely flip more than one bit.

Apart for CLIFF$_d$, where its upper bound matches that of the Fast $(1+1)$ EA$_\beta$, the new symmetric heavy-tailed operator performs asymptotically better, or at least as well as all the alternative operators discussed in this paper while allowing a more robust behaviour for escaping from the local optima of the JUMP function compared to the Fast Opt-IA$_\gamma$. A summary of the performance of all the considered operators and algorithms is provided in Table I.

We conclude with a note regarding the reasons why FCM is not strictly necessary with the operators introduced in this paper. The original HMP without FCM always flips a linear number of bits, leading to exponential expected runtimes on any function of interest [5], and requiring the use of FCM to stop if a constructive mutation is identified along the way to avoid this. The probabilistic evaluation distribution introduced in this paper allows for speed-ups during the hillclimbing phase essentially for two reasons: (1) that it allows to evaluate solutions after the first bit-flip with constant probability, and (2) that it does not make too many evaluations afterwards. Hence, the FCM scheme may be replaced by a hypermutation that simply determines the number of bits to flip according to the provided parabolic distribution, flips them all and performs only one evaluation per operation. In particular, such a choice, is much more convenient for escaping local optima with ageing, where the use of only one evaluation per mutation makes the resulting algorithm more efficient at the job (as long as the probability of flipping exactly one bit is high enough).

VIII. CONCLUSION

Due to recent analyses of increasingly realistic evolutionary algorithms, higher mutation rates than previously recommended, or than those used as a rule of thumb, are gaining significant interest in the evolutionary computation community [14], [38], [40], [45].

Such high mutation rates are naturally present in artificial immune systems. However, previous work has highlighted serious drawbacks of the hypermutation operators traditionally used in the AIS field. Firstly, while they allow to escape from local optima faster than the standard bit mutations (SBM) used by evolutionary algorithms, they do so at the expense of often being a linear factor slower at hillclimbing in the exploitation phases of the search [5], [6], [8]. Secondly, the ‘hypermutations with mutation potential’ (HMP) operators used in Opt-IA cancel out the power of the ageing operator to escape from local optima by accepting solutions of lower quality. We have presented an alternative HMP operator, FCM$_\gamma$, that provably removes these drawbacks and we have rigorously shown, for several significant benchmark problems from the literature and for classical problems from combinatorial optimisation, that it maintains the exploration characteristics of the traditional operators while outperforming them by up to linear factor speed-ups in the exploitation phases. These speed-ups at hillclimbing allow them to quickly provide feasible solutions, and high quality approximations for the NP-Hard PARTITION and VERTEX COVER problems a linear factor faster than the HMP operators traditionally used in the literature. A careful comparison with other *fast* mutation operators from the literature confirms the validity of our proposed *fast* hypermutation operators.

The main modification that allows to achieve the presented improvements over the standard static HMP with FCM is to sample the solution after the i-th bit-flip stochastically rather than deterministically. Importantly, by using a symmetric power-law distribution, we have also shown how it is possible to avoid using the FCM mechanism altogether and just evaluate one search point per hypermutation. This was probably the originally desired behaviour for the hypermutation operator of Opt-IA. However, the standard static HMP is inefficient for any function with up to a polynomial number of optima without the use of FCM [5]. Furthermore, the power-law distribution allows the *fast* HMP operators to work in harmony with ageing to escape from local optima by accepting solutions of inferior quality. This behaviour was not possible with the original

TABLE I: Expected runtimes of the standard (1+1) EA versus various hypermutation based algorithms. The best asymptotic expected runtimes for each problem are in **bold** font.
*: The asymptotic expected runtimes for obtaining a $(1+\epsilon)$ approximation for any constant ϵ.
**: The expected time to find a feasible vertex cover using the node based representation.
***: The expected time to find a 2-approximation for vertex cover using the edge representation on a graph with m edges.
† : Holds only for gap sizes in the order of $\Omega(n)$ and at most $n(\frac{1}{4}-\epsilon)$ for some constant $\epsilon > 0$.
‡: The expected runtime is obtained for $\lambda = 1$ and $\gamma = \Omega(1/\log n)$.
§: Optimal runtime obtained when $\gamma = 1/(n \log^2 n)$. The same expected runtime can be obtained by the β-algorithms for $\beta = \Omega(\log n)$.
§§: The expected runtime for the variant of the algorithm which implements hybrid ageing.

Function	(1+1) EA	$(1+1)$ IA	Fast $(1+1)$ IA$_\gamma$	Opt-IA
OneMax	$\boldsymbol{\Theta(n \log n)}$ [44]	$\Theta(n^2 \log n)$ [5]	$\boldsymbol{\Theta\left(n \log n \left(1 + \gamma \log n\right)\right)}$	$O\left(\mu \cdot \lambda \cdot n^2 \log n\right)$
LeadingOnes	$\boldsymbol{\Theta(n^2)}$ [44]	$\Theta(n^3)$ [5]	$\boldsymbol{\Theta\left(n^2 \left(1 + \gamma \log n\right)\right)}$	$O\left(\mu \cdot \lambda \cdot n^3\right)$
Trap	$\Theta(n^n)$ [44]	$\Theta(n^2 \log n)$ [5]	$\boldsymbol{O\left(n \log n \left(1 + \gamma \log n\right)\right)}$	$O\left(\mu \cdot \lambda \cdot n^2 \log n\right)$
Jump$_{d>1}$	$\Theta(n^d)$ [44]	$O(n\binom{n}{d})$ [5]	$O\left((\min\{d, n-d\}/\gamma) \cdot (1 + \gamma \log n) \cdot \binom{n}{d}\right)$	$O\left(\mu \cdot \lambda \cdot n \cdot \binom{n}{d}\right)$
Cliff$_{d>1}$	$\Theta(n^d)$ [22]	$O(n\binom{n}{d})$ [5]	$O\left((d/\gamma) \cdot (1 + \gamma \log n) \cdot \binom{n}{d}\right)$	$O(n \cdot \binom{n}{d})$
HiddenPath	$n^{\Omega(n)}$ [5]	$n^{\Omega(\log n)}$ [5]	$n^{\Omega(\log n)}$	$O(\tau \mu n + \mu n^{7/2})^\ddagger$ [5]
Partition*	$n^{\Omega(n)}$ [27]	$O(n^3)$ [9]	$\boldsymbol{O(n^2 \cdot (1 + \gamma \log n))}$	$O(\mu \cdot \lambda \cdot n^3)$
Vertex Cover**	$\boldsymbol{\Theta(n \log n)}$ [36]	$\Theta(n^2 \log n)$	$\boldsymbol{\Theta\left(n \log n \left(1 + \gamma \log n\right)\right)}$	$O(\mu \cdot \lambda \cdot n^2 \log n)$
Vertex Cover***	$\boldsymbol{\Theta(m \log m)}$ [36]	$\Theta(m^2 \log m)$	$\boldsymbol{\Theta\left(m \log m \left(1 + \gamma \log m\right)\right)}$	$O(\mu \cdot \lambda \cdot m^2 \log m)$

Function	Fast Opt-IA$_\gamma$	Fast $(1+1)$ EA$_\beta$	Fast (1+1) IA$_\beta$
OneMax	$\boldsymbol{O\left(\mu \cdot \lambda \cdot n \log n(1 + \gamma \cdot \log n)\right)}$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{O(n \log n)}$
LeadingOnes	$\boldsymbol{O\left(\mu \cdot \lambda \cdot n^2(1 + \gamma \cdot \log n)\right)}$	$\boldsymbol{O(n^2)}$	$\boldsymbol{O(n^2)}$
Trap	$\boldsymbol{O\left(\mu \cdot \lambda \cdot n \log n(1 + \gamma \cdot \log n)\right)}$	$O(n^\beta)$	$\boldsymbol{O(n \log n)}$
Jump$_{d>1}$	$O\left(\mu \cdot \lambda \cdot (\min\{d, n-d\}/\gamma) \cdot (1 + \gamma \log n) \cdot \binom{n}{d}\right)$	$O(d^\beta \binom{n}{d})$	$\boldsymbol{O\left((\min\{d, n-d\})^\beta \binom{n}{d}\right)}$
Cliff$_{d>1}$	$\boldsymbol{O\left(\frac{\mu \cdot \lambda \cdot \tau \cdot n^2}{d^2} + n \log n\right)^\S}$	$O(\tau \cdot n^{3/2})^{\dagger\S\S}$.	$O(\tau \cdot n^{3/2})^{\dagger\S\S}$.
HiddenPath	$O(\tau \mu n + \mu n^{5/2} \log n)^\ddagger$	$\boldsymbol{O(\tau n + n^{5/2})^{\S\S}}$	$\boldsymbol{O(\tau n + n^{5/2})^{\S\S}}$
Partition*	$\boldsymbol{O(\mu \cdot \lambda \cdot n^2(1 + \gamma \log n))}$	$\boldsymbol{O(n^2)}$	$\boldsymbol{O(n^2)}$
Vertex Cover**	$\boldsymbol{\Theta(\mu \cdot \lambda \cdot n \log n(1 + \gamma \log n))}$	$\boldsymbol{O(n \log n)}$	$\boldsymbol{\Theta(n \log n)}$
Vertex Cover***	$\boldsymbol{O(\mu \cdot \lambda \cdot m \log m(1 + \gamma \log m))}$	$\boldsymbol{\Theta(m \log m)}$	$\boldsymbol{\Theta(m \log m)}$

static HMP, thus considerably limiting the power of the Opt-IA algorithm where both operators are employed.

We point out that while the presented operators naturally fit within AISs, there is no reason to believe that they should not also be effective if employed within any randomised search heuristic, including EAs. Since the optimal values for the distribution parameters γ and β are different in the exploitation and the exploration phases, future work may consider an adaptation of the parameters to automatically allow them to increase and decrease throughout the run [46]–[48]. Furthermore, the performance of the proposed operators should be evaluated experimentally for classical combinatorial optimisation problems, complementing the theoretical analyses of the worst-case performance, and for real-world applications.

Acknowledgement: The research leading to these results has received funding from the EPSRC under grant agreement no EP/M004252/1 and 2020 University of Sheffield Postgraduate Research Student Publication Scholarship scheme.

References

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

[2] L. N. de Castro and F. J. Von Zuben. Learning and optimization using the clonal selection principle. *IEEE Trans. Evol. Comp.*, 6(3):239–251, 2002.

[3] J. Kelsey and J. Timmis. Immune inspired somatic contiguous hypermutation for function optimisation. In *Proc. of GECCO 2003*, pages 207–218, 2003.

[4] V. Cutello, G. Nicosia, M. Pavone, and J. Timmis. An immune algorithm for protein structure prediction on lattice models. *IEEE Trans. Evol. Comp.*, 11(1):101–117, 2007.

[5] D. Corus, P. S. Oliveto, and D. Yazdani. When hypermutations and ageing enable artificial immune systems to outperform evolutionary algorithms. *Theor. Comp. Sci.*, 832:166–185, 2020.

[6] T. Jansen and C. Zarges. Analyzing different variants of immune inspired somatic contiguous hypermutations. *Theor. Comp. Sci.*, 412(6):517 – 533, 2011.

[7] T. Jansen and C. Zarges. Computing longest common subsequences with the B-Cell Algorithm. In *Proc. of ICARIS 2012*, pages 111–124, 2012.

[8] T. Jansen, P. S. Oliveto, and C. 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.

[9] D. Corus, P. S. Oliveto, and D. Yazdani. Artificial immune systems can find arbitrarily good approximations for the NP-hard number partitioning problem. *Artificial Intelligence*, 247:180–196, 2019.

[10] D. Corus, J. He, T. Jansen, P. S. Oliveto, D. Sudholt, and C. Zarges. On easiest functions for mutation operators in bio-inspired optimisation. *Algorithmica*, 78(2):714–740, 2016.

[11] D. Corus, P. S. Oliveto, and D. Yazdani. On inversely proportional hypermutations with mutation potential. In *Proc. of GECCO 2019*, pages 215–223, 2019.

[12] P. S. Oliveto and D. Sudholt. On the runtime analysis of stochastic ageing mechanisms. In *Proc. of GECCO 2014*, pages 113–120, 2014.
[13] P. K. Lehre and C. Witt. Black-box search by unbiased variation. *Algorithmica*, 64(4):623–642, Dec 2012.
[14] B. Doerr, H. P Le, R. Makhmara, and T. D. Nguyen. Fast genetic algorithms. In *Proc. of GECCO 2017*, pages 777–784, 2017.
[15] T. Friedrich, A. Göbel, F. Quinzan, and M. Wagner. Heavy-tailed mutation operators in single-objective combinatorial optimization. In *Proc. of PPSN XV*, pages 134–145, 2018.
[16] T. Friedrich, F. Quinzan, and M. Wagner. Escaping large deceptive basins of attraction with heavy-tailed mutation operators. In *Proc. of GECCO 2018*, pages 293–300, 2018.
[17] D. Corus, P. S. Oliveto, and D. Yazdani. On the runtime analysis of the Opt-IA artificial immune system. In *Proc. of GECCO 2017*, pages 83–90, 2017.
[18] P S. Oliveto and X. Yao. Runtime analysis of evolutionary algorithms for discrete optimisation. In *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, chapter 2, pages 21–52. World Scientific, 2011.
[19] T. Jansen. *Analyzing Evolutionary Algorithms: The Computer Science Perspective*. Springer, 2013.
[20] P. K. Lehre and P. S. Oliveto. Theoretical analysis of stochastic search algorithms. In Mauricio G.C. Resende, Rafael Marti, and Panos M. Pardalos, editors, *Handbook of Heuristics*, pages 849–884. Springer, 2018.
[21] J. Jägersküpper and T. Storch. When the plus strategy outperforms the comma strategy and when not. In *Proc. of FOCI 2007*, pages 25–32, 2007.
[22] T. Paixão, J. Pérez Heredia, D. Sudholt, and B. Trubenová. Towards a runtime comparison of natural and artificial evolution. *Algorithmica*, 78(2):681–713, 2017.
[23] A. Lissovoi, P. S. Oliveto, and J. A. Warwicker. On the time complexity of algorithm selection hyper-heuristics for multimodal optimisation. In *Proc. of AAAI 2019*, pages 2322–2329, 2019.
[24] M. R Garey and D. S. Johnson. *Computers and Intractability: A Guide to the Theory of NP-Completeness*. W. H. Freeman, 1979.
[25] M. Pinedo. *Scheduling: theory, algorithms, and systems*. Prentice-Hall, 5th edition, 2016.
[26] B. Hayes. The easiest hard problem. *American Scientist*, 90:113–117, 2002.
[27] C. Witt. Worst-case and average-case approximations by simple randomized search heuristics. In *Proc. of STACS 2005*, pages 44–56, 2005.
[28] M. R. Fellows, L. Jaffke, A. I. Kiraly, F. A. Rosamond, and M. Weller. What is known about vertex cover kernelization? In *Adventures Between Lower Bounds and Higher Altitudes*, pages 330–356. springer, 2018.
[29] L. Gottlieb, A. Kontorovich, and R. Krauthgame. Efficient classification for metric data. *IEEE Trans. Info. Theor.*, 60:5750–5759, 2014.
[30] T. M. K. Cheng, Y. Lu, M. Vendruscolo, and T. L. Blundell. Prediction by graph theoretic measures of structural effects in proteins arising from non-synonymous single nucleotide polymorphisms. *PLoS Computational Biolog*, 4(7):1–9, 2008.
[31] J. Cong and M. L. Smith. A parallel bottom-up clustering algorithm with applications to circuit partitioning in vlsi design. In *Proc of 30th DAC*, pages 755–760, 1993.
[32] I. Hamzaoglu and J. H. Patel. Test set compaction algorithms for combinational circuits. In *Proc. of ICCAD 1998*, pages 283–289, 1998.
[33] S. Khuri and T. Bäck. An evolutionary heuristic for the minimum vertex cover problem. In *KI-94 Workshops (Extended abstracts)*, pages 86–90, 1994.
[34] P. S. Oliveto, J. He, and X. Yao. Analysis of the $(1+1)$ EA for finding approximate solutions to vertex cover problems. *IEEE Trans. Evol. Comp.*, 13(5):1006–1029, 2009.
[35] T. Friedrich, J. He, N. Hebbinghaus, F. Neumann, and C Witt. Approximating covering problems by randomized search heuristics using multi-objective models. *Evolutionary Computation*, 18(4):617–633, 2010.
[36] T. Jansen, P. S Oliveto, and C. Zarges. Approximating vertex cover using edge-based representations. In *Proc. of FOGA 2013*, pages 87–96, 2013.
[37] D. Corus, D. Dang, A. V. Eremeev, and P. K. Lehre. Level-based analysis of genetic algorithms and other search processes. *IEEE Trans. Evol. Comp.*, 22(5):707–719, 2017.
[38] D Corus and P.S. Oliveto. Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. *IEEE Trans. Evol. Comp.*, 22(5):720–732, 2017.
[39] D. Corus and P. S. Oliveto. On the benefits of populations on the exploitation speed of standard steady-state genetic algorithms. *Algorithmica*, 82:3676–3706, 2020.
[40] D. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, and A. M. Sutton. Escaping local optima using crossover with emergent diversity. *IEEE Trans. Evol. Comp.*, 22(3):484–497, 2018.
[41] D. Whitley. The genitor algorithm and selection pressure: why rank-based allocation of reproductive trials is best. In *Proc. of ICGA 1989*, pages 116–121, 1989.
[42] D. Corus, A. Lissovoi, P.S. Oliveto, and C. Witt. On steady-state evolutionary algorithms and selective pressure: Why inverse rank-based allocation of reproductive trials is best. *ACM Transactions on Evolutionary Learning and Optimization*, to appear, 2021. arXiv:2103.10394.
[43] C. Witt. Runtime Analysis of the $(\mu+1)$ EA on Simple Pseudo-Boolean Functions. *Evolutionary Computation*, 14(1):65–86, 2006.
[44] S. Droste, T. Jansen, and I. Wegener. On the analysis of the (1+ 1) evolutionary algorithm. *Theor. Comp. Sci.*, 276(1-2):51–81, 2002.
[45] P. S. Oliveto, P. K. Lehre, and F. Neumann. Theoretical analysis of rank-based mutation-combining exploration and exploitation. In *Proc. of CEC 2009*, pages 1455–1462, 2009.
[46] B. Doerr, A. Lissovoi, P. S. Oliveto, and J. A. Warwicker. On the runtime analysis of selection hyper-heuristics with adaptive learning periods. In *Proc. of GECCO 2018*, pages 1015–1022, 2018.
[47] B. Doerr and C. Doerr. Optimal static and self-adjusting parameter choices for the $(1+(\lambda,\lambda))$ genetic algorithm. *Algorithmica*, 80:1658–1709, 2018.
[48] A. Lissovoi, P. S. Oliveto, and J. A. Warwicker. How the duration of the learning period affects the performance of random gradient selection hyper-heuristics. In *Proc. of AAAI 2020*, pages 2376–2383, 2020.

Dogan Corus is currently a visiting academic at Kadir Has University computer engineering department. He received his PhD degree from University of Nottingham. He was formerly employed at The University of Sheffield as a research associate. His research covers bio-inspired computation techniques including artificial immune systems and evolutionary algorithms, and has particular focus on the provable mathematical statements on their performances.

Pietro S. Oliveto received the Laurea degree in computer science from the University of Catania, Catania, Italy, in 2005, and the Ph.D. degree from the University of Birmingham, Birmingham, U.K. in 2009. He is Head of the Algorithms group at the University of Sheffield, Sheffield, U.K. His current research interests include the performance analysis of bio-inspired computation techniques, covering evolutionary algorithms, genetic programming, artificial immune systems, algorithm configuration and hyper-heuristics. Dr. Oliveto is an Associate Editor of the IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, and a member of the EPSRC Peer Review College. He has been an EPSRC Ph.D.$^{+}$ Fellow and an EPSRC Post-Doctoral Fellow at Birmingham and Vice-Chancellor's Fellow and EPSRC Early Career Fellow at Sheffield.

Donya Yazdani received her Ph.D. degree in Computer Science from the University of Sheffield, Sheffield, UK, in 2020, with a thesis on the time complexity analysis of artificial immune systems for combinatorial optimization. Her current research interests include theoretical analysis of evolutionary algorithms, combinatorial optimization, and dynamic optimization problems. She is currently a lecturer at the Department of Computer Science, Aberystwyth University, Aberystwyth, UK.