Standard Steady State Genetic Algorithms Can Hillclimb Faster than Mutation-only Evolutionary Algorithms

Dogan Corus, *Member, IEEE,* and Pietro S. Oliveto, *Senior Member, IEEE,*

***Abstract*—Explaining to what extent the real power of genetic algorithms lies in the ability of crossover to recombine individuals into higher quality solutions is an important problem in evolutionary computation. In this paper we show how the interplay between mutation and crossover can make genetic algorithms hillclimb faster than their mutation-only counterparts. We devise a Markov Chain framework that allows to rigorously prove an upper bound on the runtime of standard steady state genetic algorithms to hillclimb the ONEMAX function. The bound establishes that the steady-state genetic algorithms are 25% faster than all standard bit mutation-only evolutionary algorithms with static mutation rate up to lower order terms for moderate population sizes. The analysis also suggests that larger populations may be faster than populations of size 2. We present a lower bound for a greedy (2+1) GA that matches the upper bound for populations larger than 2, rigorously proving that 2 individuals cannot outperform larger population sizes under greedy selection and greedy crossover up to lower order terms. In complementary experiments the best population size is greater than 2 and the greedy genetic algorithms are faster than standard ones, further suggesting that the derived lower bound also holds for the standard steady state (2+1) GA.**

I. INTRODUCTION

Genetic algorithms (GAs) rely on a population of individuals that simultaneously explore the search space. The main distinguishing features of GAs from other randomised search heuristics is their use of a population and crossover to generate new solutions. Rather than slightly modifying the current best solution as in more traditional heuristics, the idea behind GAs is that new solutions are generated by recombining individuals of the current population (i.e., crossover). Such individuals are selected to reproduce probabilistically according to their fitness (i.e., reproduction). Occasionally, random mutations may slightly modify the offspring produced by crossover. The original motivation behind these mutations is to avoid that some genetic material may be lost forever, thus allowing to avoid premature convergence [19], [17]. For these reasons the GA community traditionally regards crossover as the main search operator while mutation is considered a *"background operator"* [17], [1], [20] or a *"secondary mechanism of genetic adaptation"* [19].

Explaining when and why GAs are effective has proved to be a non-trivial task. Schema theory and its resulting building block hypothesis [19] were devised to explain such working principles. However, these theories did not allow to rigorously characterise the behaviour and performance of GAs. The hypothesis was disputed when a class of functions (i.e., Royal Road), thought to be ideal for GAs, was designed and experiments revealed that the simple (1+1) EA was more efficient [28], [21].

Runtime analysis approaches have provided rigorous proofs that crossover may indeed speed up the evolutionary process of GAs in ideal conditions (i.e., if sufficient diversity is available in the population). The JUMP function was introduced by Jansen and Wegener as a first example where crossover considerably improves the expected runtime compared to mutation-only Evolutionary Algorithms (EAs) [23]. The proof required an unrealistically small crossover probability to allow mutation alone to create the necessary population diversity for the crossover operator to then escape the local optimum. Dang *et al.* recently showed that the sufficient diversity, and even faster upper bounds on the runtime for not too large jump gaps, can be achieved also for realistic crossover probabilities by using diversity mechanisms [6]. Further examples that show the effectiveness of crossover have been given for both artificially constructed functions and standard combinatorial optimisation problems (see the next section for an overview).

Excellent hillclimbing performance of crossover based GAs has been also proved. B. Doerr et al. proposed a (1+(λ,λ)) GA which optimises the ONEMAX function in $\Theta(n\sqrt{\log\log\log(n)/\log\log(n)})$ fitness evaluations (i.e., runtime) [8], [9]. Since the unbiased unary black box complexity of ONEMAX is $\Omega(n\log n)$ [25], the algorithm is asymptotically faster than any unbiased mutation-only EA. Furthermore, the algorithm runs in linear time when the population size is self-adapted throughout the run [7]. Through this work, though, it is hard to derive conclusions on the working principles of standard GAs because these are very different compared to the (1+(λ,λ)) GA in several aspects. In particular, the (1+(λ,λ)) GA was especially designed to use crossover as a repair mechanism that follows the creation of new solutions via high mutation rates. This makes the algorithm work in a considerably different way compared to traditional GAs.

More traditional GAs have been analysed by Sudholt [40]. Concerning ONEMAX, he shows how (μ+λ) GAs are twice as fast as their standard bit mutation-only counterparts. As a consequence, he showed an upper bound of $(e/2)n\log n(1+o(1))$ function evaluations for a (2+1) GA versus the $en\log n(1-o(1))$ function evaluations required

D. Corus and P. S. Oliveto are with the Rigorous Research team, Algorithms group, Department of Computer Science, University of Sheffield, Regent Court, 211 Portobello, S1 4DP, Sheffield, UK. e-mail: {d.corus,p.oliveto}@sheffield.ac.uk., webpage: www.dcs.shef.ac.uk/people/P.Oliveto/rig/ .

by any standard bit mutation-only EA [39], [42]. This bound further reduces to $1.19n \ln n \pm O(n \log \log n)$ if the optimal mutation rate is used (i.e., $(1+\sqrt{5})/2 \cdot 1/n \approx 1.618/n$). However, the analysis requires that diversity is artificially enforced in the population by breaking ties always preferring genotypically different individuals. This mechanism ensures that once diversity is created on a given fitness level, it will never be lost unless a better fitness level is reached, giving ample opportunities for crossover to exploit this diversity.

Recently, it has been shown that it is not necessary to enforce diversity for standard steady state GAs to outperform standard bit mutation-only EAs [5]. In particular, the JUMP function was used as an example to show how the interplay between crossover and mutation may be sufficient for the emergence of the necessary diversity to escape from local optima more quickly. Essentially, a runtime of $O(n^{k-1})$ may be achieved for any sublinear jump length $k > 2$ versus the $\Theta(n^k)$ required by standard bit mutation-only EAs.

In this paper, we show that this interplay between mutation and crossover may also speed-up the hillclimbing capabilities of steady state GAs without the need of enforcing diversity artificially. In particular, we consider a standard steady state (μ+1) GA [36], [17], [35] and prove an upper bound on the runtime to hillclimb the ONEMAX function of $(3/4)en \log n + O(n)$ for any $\mu \geq 3$ and $\mu = o(\log n / \log \log n)$ when the standard $1/n$ mutation rate is used. Apart from showing that standard (μ+1) GAs are faster than their standard bit mutation-only counterparts up to population sizes $\mu = o(\log n / \log \log n)$, the framework provides two other interesting insights. Firstly, it delivers better runtime bounds for mutation rates that are higher than the standard $1/n$ rate. The best upper bound of $0.72en \log n + O(n)$ is achieved for c/n with $c = \frac{1}{2}\left(\sqrt{13}-1\right) \approx 1.3$. Secondly, the framework provides a larger upper bound, up to lower order terms, for the (2+1) GA compared to that of any $\mu \geq 3$ and $\mu = o(\log n / \log \log n)$. The reason for the larger constant in the leading term of the runtime is that, for populations of size 2, there is always a constant probability that any selected individual takes over the population in the next generation. This is not the case for population sizes larger than 2.

To shed light on the exact runtime for population size $\mu = 2$ we present a lower bound analysis for a greedy genetic algorithm, which we call $(2+1)_S$ GA, that always selects individuals of highest fitness for crossover and always successfully recombines them if their Hamming distance is greater than 2. This algorithm is similar to the one analysed by Sudholt [40] to allow the derivation of a lower bound, with the exception that we do not enforce any diversity artificially and that our crossover operator is slightly less greedy (i.e., in [40] crossover always recombines correctly individuals also when the Hamming distance is exactly 2). Our analysis delivers a matching lower bound for all mutation rates c/n, where c is a constant, for the greedy $(2+1)_S$ GA (thus also $(3/4)en \log n + O(n)$ and $0.72en \log n + O(n)$ respectively for mutation rates $1/n$ and $1.3/n$). This result rigorously proves that, under greedy selection and semi-greedy crossover, the (2+1) GA cannot outperform any (μ+1) GA with $\mu \geq 3$ and $\mu = o(\log n / \log \log n)$.

We present some experimental investigations to shed light on the questions that emerge from the theoretical work. In the experiments we consider the commonly used parent selection that chooses uniformly at random from the population with replacement (i.e., our theoretical upper bounds hold for a larger variety of parent selection operators). We first compare the performance of the standard steady state GAs against the fastest standard bit mutation-only EA with fixed mutation rate (i.e., the (1+1) EA [39], [42]) and the GAs that have been proved to outperform it. The experiments show that the speedups over the (1+1) EA occur already for small problem sizes n and that population sizes larger than 2 are faster than the standard (2+1) GA. Furthermore, the greedy $(2+1)_S$ GA indeed appears to be faster than the standard (2+1) GA[1], further suggesting that the theoretical lower bound also holds for the latter algorithm. Finally, experiments confirm that larger mutation rates than $1/n$ are more efficient. In particular, better runtimes are achieved for mutation rates that are even larger than the ones that minimise our theoretical upper bound (i.e., c/n with $1.5 \leq c \leq 1.6$ versus the $c = 1.3$ we have derived mathematically; interestingly this experimental rate is similar to the optimal mutation rate for OneMax of the algorithm analysed in [40]). These theoretical and experimental results seem to be in line with those recently presented for the same steady state GAs for the JUMP function [5], [6]: higher mutation rates than $1/n$ are also more effective on JUMP.

The rest of the paper is structured as follows. In the next section we briefly review previous related works that consider algorithms using crossover operators. In Section III we give precise definitions of the steady state (μ+1) GA and of the ONEMAX function. In Section IV we present the Markov Chain framework that we will use for the analysis of steady state elitist GAs. In Section V we apply the framework to analyse the (μ+1) GA and present the upper bound on the runtime for any $3 \leq \mu = o(\log n / \log \log n)$ and mutation rate c/n for any constant c. In Section VI we present the matching lower bound on the runtime of the greedy $(2+1)_S$ GA. In Section VII we present our experimental findings. In the Conclusion we present a discussion and open questions for future work. Separate supplementary files contain an appendix of omitted proofs due to space constraints and a complete version of the paper including all the proofs.

II. RELATED WORK

The first rigorous groundbreaking proof that crossover can considerably improve the performance of EAs was given by Jansen and Wegener for the (μ+1) GA with an unrealistically low crossover probability [23]. A series of following works on the analysis of the JUMP function have made the algorithm characteristics increasingly realistic [6], [24]. Today it has been rigorously proved that the standard steady state (μ+1) GA with realistic parameter settings does not require artificial diversity enforcement to outperform its standard bit mutation-only counterpart to escape the plateau of local optima of the JUMP function [5].

[1]We thank an anonymous reviewer for pointing out that this is not obvious.

Proofs that crossover may make a difference between polynomial and exponential time for escaping local optima have also been available for some time [37], [21]. The authors devised example functions where, if sufficient diversity was enforced by some mechanism, then crossover could efficiently combine different individuals into an optimal solution. Mutation, on the other hand required a long time because of the great Hamming distance between the local and global optima. The authors chose to call the artificially designed functions *Real Royal Road* functions because the Royal Road functions devised to support the building block hypothesis had failed to do so [29]. The Real Royal Road functions, though, had no resemblance with the schemata structures required by the building block hypothesis.

The utility of crossover has also been proved for less artificial problems such as coloring problems inspired by the Ising model from physics [38], computing input-output sequences in finite state machines [26], shortest path problems [14], vertex cover [30] and multi-objective optimization problems [33]. The above works show that crossover allows to escape from local optima that have large basins of attraction for the mutation operator. Hence, they establish the usefulness of crossover as an operator to enchance the exploration capabilities of the algorithm.

The interplay between crossover and mutation may produce a speed-up also in the exploitation phase, for instance when the algorithm is hillclimbing. Research in this direction has recently appeared. The design of the (1+(λ, λ)) GA was theoretically driven to beat the $\Omega(n \ln n)$ lower bound of all unary unbiased black box algorithms. Since the dynamics of the algorithm differ considerably from those of standard GAs, it is difficult to achieve more general conclusions about the performance of GAs from the analysis of the (1+(λ, λ)) GA. From this point of view the work of Sudholt is more revealing when he shows that any standard $(\mu + \lambda)$ GA outperforms its standard bit mutation-only counterpart for hillclimbing the ONEMAX function [40]. The only caveat is that the selection stage enforces diversity artificially, similarly to how Jansen and Wegener had enforced diversity for the Real Royal Road function analysis. In this paper we rigorously prove that it is not necessary to enforce diversity artificially for standard-steady state GAs to outperform their standard bit mutation-only counterpart.

III. PRELIMINARIES

We will analyse the runtime (i.e., the expected number of fitness function evaluations before an optimal search point is found) of a steady state genetic algorithm with population size μ and offspring size 1 (Algorithm 1). In steady state GAs the entire population is not changed at once, but rather a part of it. In this paper we consider the most common option of creating one new solution per generation [36], [35]. Rather than restricting the algorithm to the most commonly used uniform selection of two parents, we allow more flexibility to the choice of which parent selection mechanism is used. This approach was also followed by Sudholt for the analysis of the (μ+1) GA with diversity [40]. In each generation the algorithm picks two parents from its population with replacement using a selection operator that satisfies the following condition.

Algorithm 1: (μ+1) GA [36], [17], [35], [5]

```
P ← μ individuals, uniformly at random from {0,1}^n;
repeat
    Select x, y ∈ P with replacement using an operator abiding (1);
    z ← Uniform crossover with probability 1/2 (x, y);
    Flip each bit in z with probability c/n;
    P ← P ∪ {z};
    Choose one element from P with lowest fitness and remove it from P, breaking ties at random;
until termination condition satisfied;
```

$$\forall x, y : \; f(x) \geq f(y) \implies \Pr(\text{select } x) \geq \Pr(\text{select } y). \quad (1)$$

The condition allows to use most of the popular parent selection mechanisms with replacement such as fitness proportional selection, rank selection or the one commonly used in steady state GAs, i.e., uniform selection [17]. Afterwards, uniform crossover between the selected parents (i.e., each bit of the offspring is chosen from each parent with probability $1/2$) provides an offspring to which standard bit mutation (i.e., each bit is flipped with with probability c/n) is applied. The best μ among the $\mu + 1$ solutions are carried over to the next generation and ties are broken uniformly at random.

In the paper we use the standard convention for naming steady state algorithms: the (μ+1) EA differs from the (μ+1) GA by only selecting one individual per generation for reproduction and applying standard bit mutation to it (i.e., no crossover). Otherwise the two algorithms are identical.

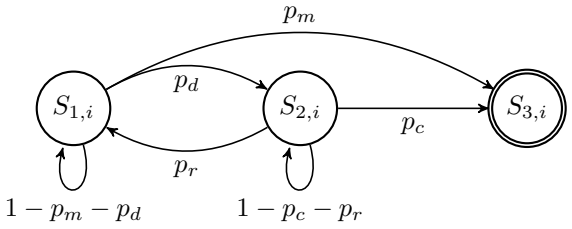
We will analyse Algorithm 1 for the well-studied ONEMAX function that is defined on bitstrings $x \in \{0,1\}^n$ of length n and returns the number of 1-bits in the string: $\text{ONEMAX}(x) = \sum_{i=1}^{n} x_i$. Here x_i is the ith bit of the solution $x \in \{0,1\}^n$. The ONEMAX benchmark function is very useful to assess the hillclimbing capabilities of a search heuristic. It displays the characteristic function optimisation property that finding improving solutions becomes harder as the algorithm approaches the optimum. The problem is the same as that of identifying the hidden solution of the Mastermind game where we assume for simplicity that the target string is the one of all 1-bits. Any other target string $z \in \{0,1\}^n$ may also be used without loss of generality. If a bitstring is used, then ONEMAX is equivalent to Mastermind with two colours [16]. This can be generalised to many colours if alphabets of greater size are used [12], [11].

IV. MARKOV CHAIN FRAMEWORK

The recent analysis of the (μ+1) GA for the JUMP function shows that the interplay between crossover and mutation may create the diversity required for crossover to decrease the expected time to jump towards the optimum [5]. At the heart of the proof is the analysis of a random walk on the

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2745715, IEEE Transactions on Evolutionary Computation

Fig. 1. Markov Chain for fitness level i.



number of diverse individuals on the local optima of the function. The analysis delivers improved asymptotic expected runtimes of the (μ+1) GA over mutation-only EAs only for population sizes $\mu = \omega(1)$. This happens because, for larger population sizes, it takes more time to lose diversity once created, hence crossover has more time to exploit it. For ONEMAX the technique delivers worse asymptotic bounds for population sizes $\mu = \omega(1)$ and an $O(n \ln n)$ bound for constant population size. Hence, the techniques of [5] cannot be directly applied to show a speed-up of the (μ+1) GA over mutation-only EAs and a careful analysis of the leading constant in the runtime is necessary. In this section we present the Markov chain framework that we will use to obtain the upper bounds on the runtime of the elitist steady state GAs. We will afterwards discuss how this approach builds upon and generalises Sudholt's approach in [40].

The ONEMAX function has $n + 1$ distinct fitness values. We divide the search space into the following canonical fitness levels [22], [32]:

$$L_i = \{x \in \{0,1\}^n | \textsc{OneMax}(x) = i\}.$$

We say that a population is in fitness level i if and only if its best solution is in level L_i.

We use a Markov chain (MC) for each fitness level i to represent the different states the population may be in before reaching the next fitness level. The MC depicted in Fig. 1 distinguishes between states where the population has no diversity (i.e., all individuals have the same genotype), hence crossover is ineffective, and states where diversity is available to be exploited by the crossover operator. The MC has one absorbing state and two transient states. The first transient state $S_{1,i}$ is adopted if the whole population consists of copies of the same individual at level i (i.e., all the individuals have the same genotype). The second state $S_{2,i}$ is reached if the population consists of μ individuals in fitness level i and at least two individuals x and y are not identical. The second transient state $S_{2,i}$ differs from the state $S_{1,i}$ in having diversity which can be exploited by the crossover operator. $S_{1,i}$ and $S_{2,i}$ are mutually accessible from each other since the diversity can be introduced at state $S_{1,i}$ via mutation with some probability p_d and can be lost at state $S_{2,i}$ with some relapse probability p_r when copies of a solution take over the population.

The absorbing state $S_{3,i}$ is reached when a solution at a better fitness level is found, an event that happens with probability p_m when the population is at state $S_{1,i}$ and with probability p_c when the population is at state $S_{2,i}$. We pessimistically assume that in $S_{2,i}$ there is always only one single individual with a different genotype (i.e., with more than one distinct individual, p_c would be higher and p_r would be zero). Formally when $S_{3,i}$ is reached the population is no longer in level i because a better fitness level has been found. However, we will bound the expected time to reach the absorbing state for the next level only when the whole population has reached it (or a higher level). We do this because we assume that initially all the population is in level i when calculating the transition probabilities in the MC for each level i. This implies that bounding the expected times to reach the absorbing states of each fitness level is not sufficient to achieve an upper bound on the total expected runtime. When $S_{3,i}$ is reached for the first time, the population only has one individual at the next fitness level or in a higher one. Only when all the individuals have reached level $i + 1$ (i.e., either in state $S_{1,i+1}$ or $S_{2,i+1}$) may we use the MC to bound the runtime to overcome level $i + 1$. Then the MC can be applied, once per fitness level, to bound the total runtime until the optimum is reached.

The main distinguishing aspect between the analysis presented herein and that of Sudholt [40] is that we take into account the possibility to transition back and forth (i.e., resp. with probability p_d and p_r) between states $S_{1,i}$ and $S_{2,i}$ as in standard steady state GAs (see Fig. 1). By enforcing that different genotypes on the same fitness level are kept in the population, the genetic algorithm considered in [40] has a good probability of exploiting this diversity to recombine the different individuals. In particular, once the diversity is created it will never be lost, giving many opportunities for crossover to take advantage of it. A crucial aspect is that the probability of increasing the number of ones via crossover is much higher than the probability of doing so via mutation once many 1-bits have been collected. Hence, by enforcing that once State $S_{2,i}$ is reached it cannot be left until a higher fitness level is found, Sudholt could prove that the resulting algorithm is faster compared to only using standard bit mutation. In the standard steady state GA, instead, once the diversity is created it may subsequently be lost before crossover successfully recombines the diverse individuals. This behaviour is modelled in the MC by considering the relapse probability p_r. Hence, the algorithm spends less time in state $S_{2,i}$ compared to the GA with diversity enforcement. Nevertheless, it will still spend some optimisation time in state $S_{2,i}$ where it will have a higher probability of improving its fitness by exploiting the diversity via crossover than when in state $S_{1,i}$ (i.e., no diversity) where it has to rely on mutation only. For this reason the algorithm will not be as fast for ONEMAX as the GA with enforced diversity but will still be faster than standard bit mutation-only EAs.

An interesting consequence of the possibility of losing diversity, is that populations of size greater than 2 can be beneficial. In particular the diversity (i.e., State $S_{2,i}$) may be completely lost in the next step when there is only one diverse individual left in the population. When this is the case, the relapse probability p_r decreases with the population size μ because the probability of selecting the diverse individual for removal is $1/\mu$. Furthermore, for population size $\mu = 2$ there is a positive probability that diversity is lost in every generation by either of the two individuals taking over, while

for larger population sizes this is not the case. As a result our MC framework analysis will deliver a better upper bound for $\mu > 2$ compared to the bound for $\mu = 2$. This interesting insight into the utility of larger populations could not be seen in the analysis of [40] because there, once the diversity is achieved, it cannot be lost.

We first concentrate on the expected absorbing time of the MC. Afterwards we will calculate the takeover time before we can transition from one MC to the next. Since it is not easy to derive the exact transition probabilities, a runtime analysis is considerably simplified by using bounds on these probabilities. The main result of this section is stated in the following theorem that shows that we can use lower bounds on the transition probabilities moving in the direction of the absorbing state (i.e., p_m, p_d and p_c) and an upper bound on the probability of moving in the opposite direction to no diversity (i.e., p_r) to derive an upper bound on the expected absorbing time of the Markov chain. In particular, we define a Markov chain M' that uses the bounds on the exact transition probabilities and show that its expected absorbing time is greater than the absorbing time of the original chain. Hereafter, we drop the level index i for brevity and use $E[T_1]$ and $E[T_2]$ instead of $E[T_{1,i}]$ and $E[T_{2,i}]$ (Similarly, S_1 will denote state $S_{1,i}$).

Theorem 1. *Consider two Markov chains M and M' with the topology in Figure 1 where the transition probabilities for M are p_c, p_m, p_d , p_r and the transition probabilities for M' are p'_c, p'_m, p'_d and p'_r. Let the expected absorbing time for M be $E[T]$ and the expected absorbing time of M' starting from state S_1 be $E[T'_1]$ respectively. If*

- $p_m < p_c$
- $p'_d \leq p_d$
- $p'_r \geq p_r$
- $p'_c \leq p_c$
- $p'_m \leq p_m$

Then $E[T] \leq E[T'_1] \leq \frac{p'_c+p'_r}{p'_cp'_d+p'_cp'_m+p'_mp'_r} + \frac{1}{p'_c}$.

We first concentrate on the second inequality in the statement of the theorem which will follow immediately from the next lemma. It allows us to obtain the expected absorbing time of the MC if the exact values for the transition probabilities are known. In particular, the lemma establishes the expected times $E[T_1]$ and $E[T_2]$ to reach the absorbing state, starting from the states S_1 and S_2 respectively.

Lemma 2. *The expected times $E[T_1]$ and $E[T_2]$ to reach the absorbing state, starting from state S_1 and S_2 respectively are as follows:*

$$E[T_1] = \frac{p_c + p_r + p_d}{p_cp_d + p_cp_m + p_mp_r} \leq \frac{p_c + p_r}{p_cp_d + p_cp_m + p_mp_r} + \frac{1}{p_c}$$

$$E[T_2] = \frac{p_m + p_r + p_d}{p_cp_d + p_cp_m + p_mp_r}.$$

Before we prove the first inequality in the statement of Theorem 1, we will derive some helper propositions. We first show that as long as the transition probability of reaching the absorbing state from the state S_2 (with diversity) is greater than that of reaching the absorbing state from the state with no diversity S_1 (i.e., $p_m < p_c$), then the expected absorbing time from state S_1 is at least as large as the expected time unconditional of the starting point. This will allow us to achieve a correct upper bound on the runtime by just bounding the absorbing time from state S_1. In particular, it allows us to pessimistically assume that the algorithm starts each new fitness level in state S_1 (i.e., there is no diversity in the population).

Proposition 3. *Consider a Markov chain with the topology given in Figure 1. Let $E[T_1]$ and $E[T_2]$ be the expected absorbing times starting from state S_1 and S_2 respectively. If $p_m < p_c$, then $E[T_1] > E[T_2]$ and $E[T]$, the unconditional expected absorbing time, satisfies $E[T] \leq E[T_1]$.*

In the following proposition we show that if we overestimate the probability of losing diversity and underestimate the probability of increasing it, then we achieve an upper bound on the expected absorbing time as long as $p_m < p_c$. Afterwards, in Proposition 5 we show that an upper bound on the absorbing time is also achieved if the probabilities p_c and p_m are underestimated.

Proposition 4. *Consider two Markov chains M and M' with the topology in Figure 1 where the transition probabilities for M are p_c, p_m, p_d, p_r and the transition probabilities for M' are p_c, p_m, p'_d and p'_r. Let the expected absorbing times starting from state S_1 for M and M' be $E[T_1]$ and $E[T'_1]$ respectively. If $p'_d \leq p_d$, $p'_r \geq p_r$ and $p_m < p_c$, then $E[T_1] \leq E[T'_1]$.*

Proposition 5. *Consider two Markov chains M and M' with the topology in Figure 1 where the transition probabilities for M are p_c, p_m, p_d, p_r and the transition probabilities for M' are p'_c, p'_m, p_d and p_r. Let the expected absorbing times starting from state S_1 for M and M' be $E[T_1]$ and $E[T'_1]$ respectively. If $p'_c \leq p_c$ and $p'_m \leq p_m$, then $E[T_1] \leq E[T'_1]$.*

The propositions use that by lower bounding p_d and upper bounding p_r we overestimate the expected number of generations the population is in state S_1 compared to the time spent in state S_2. Hence, if $p_c > p_m$ we can safely use a lower bound for p_d and an upper bound for p_r and still obtain a valid upper bound on the runtime $E[T_1]$. This is rigorously shown by combining together the results of the previous propositions to prove the main result i.e., Theorem 1.

Proof of Theorem 1. Consider a third Markov chain M^* whose transition probabilities are p_c, p_m, p'_r, p'_d. Let the absorbing time of M starting from state S_1 be $E[T_1]$. In order to prove the above statement we will prove the following sequence of inequalities:

$$E[T] \leq E[T_1] \leq E[T_1^*] \leq E[T'_1].$$

According to Proposition 3, $E[T] \leq E[T_1]$ since $p_c > p_m$. According to Proposition 4 $E[T_1] \leq E[T_1^*]$ since $p'_d \leq p_d$, $p'_r \geq p_r$ and $p_c > p_m$. Finally, according to Proposition 5, $p'_c \leq p_c$ and $p'_m \leq p_m$ implies $E[T_1^*] \leq E[T'_1]$ and our proof is completed by using Lemma 2 to show that the last inequality of the statement holds. □

The algorithm may skip some levels or a new fitness level may be found before the whole population has reached the

current fitness level. Hence, by summing up the expected runtimes to leave each of the $n+1$ levels and the expected times for the whole population to takeover each level, we obtain an upper bound on the expected runtime. The next lemma establishes an upper bound on the expected time it takes to move from the absorbing state of the previous Markov chain ($S_{3,i}$) to any transient state ($S_{1,i+1}$ or $S_{2,i+1}$) of the next Markov chain. The lemma uses standard takeover arguments originally introduced in the first analysis of the (μ+1) EA for ONEMAX [41]. To achieve a tight upper bound Witt had to carefully wait for only a fraction of the population to take over a level before the next level was discovered. In our case, the calculation of the transition probabilities of the MC is actually simplified if we wait for the whole population to take over each level. Hence in our analysis the takeover time calculations are more similar to the first analysis of the (μ+1) EA with and without diversity mechanisms to takeover the local optimum of TWOMAX [18].

Lemma 6. *Let the best individual of the current population be in level i and all individuals be in level at least $i-1$. Then, the expected time for the whole population to be in level at least i is $\mathcal{O}(\mu \log \mu)$.*

The lemma shows that, once a new fitness level is discovered for the first time, it takes at most $\mathcal{O}(\mu \log \mu)$ generations until the whole population consists of individuals from the newly discovered fitness level or higher. While the absorption time of the Markov chain might decrease with the population size, for too large population sizes, the upper bound on the expected total take over time will dominate the runtime. As a result the MC framework will deliver larger upper bounds on the runtime unless the expected time until the population takes over the fitness levels is asymptotically smaller than the expected absorption time of all MCs. For this reason, our results will require population sizes of $\mu = o(\log n / \log \log n)$, to allow all fitness levels to be taken over in expected $o(n \log n)$ time such that the latter time does not affect the leading constant of the total expected runtime.

V. UPPER BOUND

In this section we use the Markov Chain framework devised in Section IV to prove that the (μ+1) GA is faster than any standard bit mutation-only ($\mu+\lambda$) EA.

In order to satisfy the requirements of Theorem 1, we first show in Lemma 7 that $p_c > p_m$ if the population is at one of the final $n/(4c(1+e^c))$ fitness levels. The lemma also shows that it is easy for the algorithm to reach such a fitness level. Afterwards we bound the transition probabilities of the MC in Lemma 8. We conclude the section by stating and proving the main result, essentially by applying Theorem 1 with the transition probabilities calculated in Lemma 8.

Lemma 7. *For the (μ+1) GA with mutation rate c/n for any constant c, if the population is in any fitness level $i > n - n/(4c(1+e^c))$, then p_c is always larger than p_m. The expected time for the (μ+1) GA to sample a solution in fitness level $n - n/(4c(1+e^c))$ for the first time is $\mathcal{O}(n\mu \log \mu)$.*

Proof. We consider the probability p_c. If two individuals on the same fitness level with non-zero Hamming distance $2d$ are selected as parents with probability p', then the probability that the crossover operator yields an improved solution is at least (see proof of Theorem 4 in [40]):

$$\begin{aligned}\Pr(X > d) =& \frac{1}{2}\left(1 - \Pr(X = d)\right) \\ =& \frac{1}{2}\left(1 - 2^{-2d}\binom{2d}{d}\right) \geq 1/4, \end{aligned} \tag{2}$$

where X is a binomial random variable with parameters $2d$ and $1/2$ which represents the number of bit positions where the parents differ and which are set to 1 in the offspring. With probability $(1-c/n)^n$ no bits are flipped and the absorbing state is reached. If any individual is selected twice as parent, then the improvement can only be achieved by mutation (i.e., with probability p_m) since crossover is ineffective. So $p_c > p'(1/4)(1-c/n)^n + (1-p')p_m$, hence if $p_m < p'(1/4)(1-c/n)^n + (1-p')p_m$ it follows that $p_m < p_c$. The condition can be simplified to $p_m < (1/4)(1-c/n)^n$ with simple algebraic manipulation. For large enough n, $(1-c/n)^n \geq 1/(1+e^c)$ and the condition reduces to $p_m < 1/(4(1+e^c))$.

Since $p_m < (n-i)c/n$ is an upper bound on the transition probability (i.e., at least one of the zero bits has to flip to increase the ONEMAX value), the condition is satisfied for $i \geq n - n/(4c(1+e^c))$. For any level $i \leq n - n/(4c(1+e^c))$, after the take over of the level occurs in $\mathcal{O}(\mu \log \mu)$ expected time, the probability of improving is at least $\Omega(1)$ due to the linear number of 0-bits that can be flipped. Hence, we can upper bound the total number of generations necessary to reach fitness level $i = n - n/(4c(1+e^c))$ by $\mathcal{O}(n\mu \log \mu)$. □

The lemma has shown that $p_c > p_m$ holds after a linear number of fitness levels have been traversed. Now, we bound the transition probabilities of the Markov chain.

Lemma 8. *Let $\mu \geq 3$. Then the transition probabilities p_d, p_c, p_r and p_m are bounded as follows:*

$$p_d \geq \frac{\mu}{(\mu+1)}\frac{i(n-i)c^2}{n^2(e^c + \mathcal{O}(1/n))}, \qquad p_c \geq \frac{\mu - 1}{2\mu^2(e^c + \mathcal{O}(1/n))},$$
$$p_r \leq \frac{(\mu-1)\big(2\mu - 1 + \mathcal{O}(1/n)\big)}{2e^c\mu^2(\mu+1)}, \quad p_m \geq \frac{c(n-i)}{n(e^c + \mathcal{O}(1/n))}.$$

Proof. We first bound the probability p_d of transitioning from the state $S_{1,i}$ to the state $S_{2,i}$. In order to introduce a new solution at level i with different genotype, it is sufficient that the mutation operator simultaneously flips one of the $n-i$ 0-bits and one of the i 1-bits while not flipping any other bit. We point out that in $S_{1,i}$, all individuals are identical, hence crossover is ineffective. Moreover, when the diverse solution is created, it should stay in the population, which occurs with probability $\mu/(\mu+1)$ since one of the μ copies of the majority individual should be removed by selection instead of the offspring. So p_d can be lower bounded as follows:

$$p_d \geq \frac{\mu}{(\mu+1)}\frac{ic}{n}\frac{(n-i)c}{n}\left(1 - \frac{c}{n}\right)^{n-2}.$$

Using the inequality $(1-1/x)^{x-1} \geq 1/e \geq (1-1/x)^x$, we now bound $\left(1-\frac{c}{n}\right)^{n-2}$ as follows:

$$\begin{aligned}\left(1-\frac{c}{n}\right)^{n-2} &\geq \left(1-\frac{c}{n}\right)^{n-1} \geq \left(1-\frac{c}{n}\right)^{n} \\ &= \left(\left(1-\frac{c}{n}\right)^{(n/c)-1}\left(1-\frac{c}{n}\right)\right)^{c} \\ &\geq \left(\frac{1}{e}\left(1-\frac{c}{n}\right)\right)^{c} \geq \frac{1}{e^c}\left(1-\frac{c^2}{n}\right),\end{aligned}$$

where in the last step we used the Bernoulli's inequality. We can further absorb the c^2/n in an asymptotic $\mathcal{O}(1/n)$ term as follows:

$$\left(1-\frac{c}{n}\right)^{n-2} \geq e^{-c} - \mathcal{O}(1/n) = \frac{1}{e^c + \mathcal{O}(1/n)}. \quad (3)$$

The bound for p_d is then,

$$p_d \geq \frac{\mu}{(\mu+1)}\frac{i(n-i)c^2}{n^2\left(e^c+\mathcal{O}(1/n)\right)}.$$

We now consider p_c. To transition from state $S_{2,i}$ to $S_{3,i}$ (i.e., p_c) it is sufficient that two genotypically different individuals are selected as parents (i.e., with probability at least $2(\mu-1)/\mu^2$), that crossover provides a better solution (i.e., with probability at least $1/4$ according to Eq. (2)) and that mutation does not flip any bits (i.e., probability $(1-c/n)^n \geq 1/\left(e^c+\mathcal{O}(1/n)\right)$ according to Eq. (3)). Therefore, the probability is

$$p_c \geq 2\frac{\mu-1}{\mu^2}\frac{1}{4}\left(1-\frac{c}{n}\right)^n \geq \frac{\mu-1}{2\mu^2(e^c+\mathcal{O}(1/n))}$$

For calculating p_r we pessimistically assume that the Hamming distance between the individuals in the population is 2 and that there is always only one individual with a different genotype. A population in state $S_{2,i}$ which has diversity, goes back to state $S_{1,i}$ when:

1) A majority individual is selected twice as parent (i.e., probability $(\mu-1)^2/\mu^2$), mutation does not flip any bit (i.e., probability $(1-c/n)^n$) and the minority individual is discarded (i.e., probability $1/(\mu+1)$).
2) Two different individuals are selected as parents, crossover chooses either from the majority individual in both bit locations where they differ (i,e., prob. 1/4) and mutation does not flip any bit (i.e., probability $(1-c/n)^n \leq 1/e^c$) or mutation must flip at least one specific bit (i.e., probability $\mathcal{O}(1/n)$). Finally, the minority individual is discarded (i.e., probability $1/(\mu+1)$).
3) A minority individual is chosen twice as parent and the mutation operator flips at least two specific bit positions (i.e., with probability $\mathcal{O}(1/n^2)$) and finally the minority individual is discarded (i.e., probability $1/(\mu+1)$).

Hence, the probability of losing diversity is:

$$\begin{aligned}p_r &\leq \frac{1}{\mu+1}\left[\frac{(\mu-1)^2}{\mu^2}\left(1-\frac{c}{n}\right)^n\right. \\ &\quad \left. + 2\frac{1}{\mu}\frac{\mu-1}{\mu}\left(\frac{1}{4}\left(1-\frac{c}{n}\right)^n + \mathcal{O}(1/n)\right) + \mathcal{O}(1/n^2)\right] \\ &\leq \frac{2(\mu-1)^2 + (\mu-1) + 4e^c(\mu-1)\mathcal{O}(1/n)}{2e^c\mu^2(\mu+1)} + \frac{\mathcal{O}(1/n^2)}{\mu+1} \\ &= \frac{(\mu-1)[2(\mu-1)+1+4e^c\mathcal{O}(1/n)] + 2e^c\mu^2\mathcal{O}(1/n^2)}{2e^c\mu^2(\mu+1)} \\ &= \frac{(\mu-1)[2(\mu-1)+1+\mathcal{O}(1/n)] + \mathcal{O}(1/n^2)}{2e^c\mu^2(\mu+1)} \\ &\leq \frac{(\mu-1)(2\mu-1+\mathcal{O}(1/n))}{2e^c\mu^2(\mu+1)}.\end{aligned}$$

In the last inequality we absorbed the $\mathcal{O}(1/n^2)$ term into the $\mathcal{O}(1/n)$ term.

The transition probability p_m from state $S_{1,i}$ to state $S_{3,i}$ is the probability of improvement by mutation only, because crossover is ineffective at state $S_{1,i}$. The number of 1-bits in the offspring increases if the mutation operator flips one of the $(n-i)$ 0-bits (i.e., with probability $c(n-i)/n$) and does not flip any other bit (i.e., with probability $(1-c/n)^{n-1} \geq \left(e^c+\mathcal{O}(1/n)\right)^{-1}$ according to Eq. (3)). Therefore, the lower bound on the probability p_m is:

$$p_m \geq \frac{c(n-i)}{n\left(e^c+\mathcal{O}(1/n)\right)}.$$

□

We are finally ready to state our main result.

Theorem 9. *The expected runtime of the (μ+1) GA with $\mu \geq 3$ and mutation rate c/n for any constant c on* ONEMAX *is:*

$$E[T] \leq \frac{3e^c n\log n}{c(3+c)} + \mathcal{O}(n\mu\log\mu).$$

For $\mu = o(\log n/\log\log n)$, the bound reduces to:

$$E[T] \leq \frac{3}{c(3+c)}e^c n\log n\,(1+o(1)).$$

Proof. We use Theorem 1 to bound $E[T_i]$, the expected time until the (μ+1) GA creates an offspring at fitness level $i+1$ or above given that all individuals in its initial population are at level i. The bounds on the transition probabilities established in Lemma 8 will be set as the exact transition probabilities of another Markov chain, M', with absorbing time larger than $E[T_i]$ (by Theorem 1). Since Theorem 1 requires that $p_c > p_m$ and Lemma 7 establishes that $p_c > p_m$ holds for all fitness levels $i > n - n/4c(1+e^c)$, we will only analyse $E[T_i]$ for $n-1 \geq i > n - n/\left(4c(1+e^c)\right)$. Recall that, by Lemma 7, level $n - n/\left(4c(1+e^c)\right)$ is reached in expected $\mathcal{O}(n\mu\log\mu)$ time.

Consider the expected absorbing time $E[T_i']$, of the Markov chain M' with transition probabilities:

$$p_d' := \frac{\mu}{(\mu+1)}\frac{i(n-i)c^2}{n^2(e^c+\mathcal{O}(1/n))}, \quad p_c' := \frac{\mu-1}{2\mu^2(e^c+\mathcal{O}(1/n))},$$
$$p_r' := \frac{(\mu-1)\left(2\mu-1+\mathcal{O}(1/n)\right)}{2e^c\mu^2(\mu+1)}, \quad p_m' := \frac{c(n-i)}{n(e^c+\mathcal{O}(1/n))}.$$

According to Theorem 1:

$$E[T_i] \leq E[T_{i,1}'] \leq \frac{p_c'+p_r'}{p_c'p_d' + p_c'p_m' + p_m'p_r'} + \frac{1}{p_c'}. \quad (4)$$

We simplify the numerator and the denominator of the first term separately. The numerator is

$$\begin{aligned} p'_c + p'_r &= \frac{\mu-1}{2\mu^2(e^c+\mathcal{O}(1/n))} + \frac{(\mu-1)\big(2\mu-1+\mathcal{O}(1/n)\big)}{2e^c\mu^2(\mu+1)} \\ &\leq \frac{\mu-1}{2\mu^2 e^c}\left(1+\frac{2\mu-1+\mathcal{O}(1/n)}{\mu+1}\right) \\ &\leq \frac{(\mu-1)[3\mu+\mathcal{O}(1/n)]}{2\mu^2 e^c(\mu+1)}. \end{aligned} \tag{5}$$

We can also rearrange the denominator $D = p'_c p'_d + p'_c p'_m + p'_m p'_r$ as follows:

$$\begin{aligned} D &= p'_c(p'_d+p'_m) + p'_m p'_r \\ &= \frac{(\mu-1)\left(\frac{\mu i(n-i)c^2}{(\mu+1)n^2(e^c+\mathcal{O}(1/n))} + \frac{c(n-i)}{n(e^c+\mathcal{O}(1/n))}\right)}{2\mu^2\left(e^c+\mathcal{O}(1/n)\right)} \\ &\quad + \frac{c(n-i)(\mu-1)\big(2\mu-1+\mathcal{O}(1/n)\big)}{n\left(e^c+\mathcal{O}(1/n)\right)2e^c\mu^2(\mu+1)} \\ &\geq \frac{(\mu-1)\left(\frac{\mu i(n-i)c^2}{(\mu+1)n^2} + \frac{c(n-i)}{n}\right)}{2\mu^2\left(e^{2c}+\mathcal{O}(1/n)\right)} \\ &\quad + \frac{c(n-i)(\mu-1)\big(2\mu-1+\mathcal{O}(1/n)\big)}{n\left(e^{2c}+\mathcal{O}(1/n)\right)2\mu^2(\mu+1)} \\ &\geq \frac{c(n-i)(\mu-1)}{2\mu^2\left(e^{2c}+\mathcal{O}(1/n)\right)}\cdot \\ &\quad \left(\frac{\mu ic}{(\mu+1)n^2} + \frac{1}{n} + \frac{2\mu-1+\mathcal{O}(1/n)}{n(\mu+1)}\right) \\ &\geq \frac{c(n-i)(\mu-1)}{2\mu^2\left(e^{2c}+\mathcal{O}(1/n)\right)}\cdot \\ &\quad \left(\frac{\mu ic+(\mu+1)n+n\big(2\mu-1+\mathcal{O}(1/n)\big)}{(\mu+1)n^2}\right) \\ &\geq \frac{c(n-i)(\mu-1)\Big(\mu ic+n\Big[3\mu+\mathcal{O}(1/n)\Big]\Big)}{2\mu^2\left(e^{2c}+\mathcal{O}(1/n)\right)(\mu+1)n^2}. \end{aligned} \tag{6}$$

Note that the term in square brackets is the same in both the numerator (i.e., Eq. (5)) and the denominator (i.e., Eq. (6)) including the small order terms in $\mathcal{O}(1/n)$ (i.e., they are identical). Let $A = [3\mu + c'/n]$, where $c' > 0$ is the smallest constant that satisfies the $\mathcal{O}(1/n)$ in the upper bound on p_r in Lemma 8. We can now put the numerator and denominator together and simplify the expression :

$$\begin{aligned} &(p'_c+p'_r)/\big(p'_c(p'_d+p'_m)+p'_m p'_r\big) \\ &\leq \frac{(\mu-1)A}{2\mu^2 e^c(\mu+1)}\cdot\frac{2\mu^2\left(e^{2c}+\mathcal{O}(1/n)\right)(\mu+1)n^2}{c(n-i)(\mu-1)(\mu ic+nA)} \\ &\leq \frac{A\left(e^{2c}+\mathcal{O}(1/n)\right)n^2}{e^c c(n-i)(\mu ic+nA)}. \end{aligned}$$

By using that $\frac{e^{2c}+\mathcal{O}(1/n)}{e^c} \leq e^c + \mathcal{O}(1/n)$, we get:

$$\begin{aligned} &\leq \frac{A\left(e^c+\mathcal{O}(1/n)\right)n^2}{c(n-i)(\mu ic+nA)} \\ &\leq e^c\frac{An^2}{c(n-i)(\mu ic+nA)} + \mathcal{O}(1/n)\frac{An^2}{c(n-i)(\mu ic+nA)}. \end{aligned}$$

The facts, $n-i \geq 1$, $A = \Omega(1)$, and $\mu, i, c > 0$ imply that, $nA + \mu ic = \Omega(n)$ and $\frac{An^2}{c(n-i)(\mu ic+nA)} = \mathcal{O}(n)$. When multiplied by the $\mathcal{O}(1/n)$ term, we get:

$$\leq \frac{e^c n}{c(n-i)}\frac{An}{(\mu ic+nA)} + \mathcal{O}(1).$$

By adding and subtracting μic to the numerator of $\frac{An}{(\mu ic+nA)}$, we obtain:

$$\leq \frac{e^c n}{c(n-i)}\left(1-\frac{\mu ic}{\mu ic+nA}\right) + \mathcal{O}(1).$$

Note that the multiplier outside the brackets, $(e^c n)/(c(n-i))$, is in the order of $\mathcal{O}\big(n/(n-i)\big)$. We now add and subtract μnc to the numerator of $-\frac{\mu ic}{\mu ic+nA}$ to create a positive additive term in the order of $\mathcal{O}\big(\mu(n-i)/n\big)$.

$$\begin{aligned} &= \frac{e^c n}{c(n-i)}\left(1-\frac{\mu nc}{\mu ic+nA}+\frac{\mu(n-i)c}{\mu ic+nA}\right)+\mathcal{O}(1) \\ &= \frac{e^c n}{c(n-i)}\left(1-\frac{\mu nc}{\mu ic+nA}\right) + \frac{e^c n}{c(n-i)}\frac{\mu(n-i)c}{\mu ic+nA} \\ &\quad + \mathcal{O}(1) = \frac{e^c n}{c(n-i)}\left(1-\frac{\mu nc}{\mu ic+nA}\right)+\mathcal{O}(\mu). \end{aligned}$$

Since $p'_c = \Omega(1/\mu)$, we can similarly absorb $1/p'_c$ into the $\mathcal{O}(\mu)$ term. After the addition of the remaining term $1/p'_c$ from Eq.(4), we obtain a valid upper bound on $E[T_i]$:

$$\begin{aligned} E[T_i] &\leq \frac{p'_c+p'_r}{p'_c p'_d + p'_c p'_m + p'_m p'_r} + \frac{1}{p'_c} \\ &\leq \frac{e^c n}{c(n-i)}\left(1-\frac{\mu nc}{\mu ic+nA}\right)+\mathcal{O}(\mu). \end{aligned}$$

In order to bound the negative term, we will rearrange its denominator (i.e., $nA + \mu ic$):

$$\begin{aligned} &n\big[3\mu+c'/n\big]+\mu = 3\mu n + c' + \mu ic \\ &\quad = 3\mu n + c' - (n-i)\mu c + \mu nc < \mu n(3+c) + c', \end{aligned}$$

where the second equality is obtained by adding and subtracting μnc. Altogether,

$$\begin{aligned} E[T_i] &\leq \frac{e^c n}{c(n-i)}\left(1-\frac{\mu nc}{\mu n(3+c)+c'}\right)+\mathcal{O}(\mu) \\ &= \frac{e^c n}{c(n-i)}\left(1-\frac{\mu nc + c'\frac{c}{3+c} - c'\frac{c}{3+c}}{\mu n(3+c)+c'}\right)+\mathcal{O}(\mu) \\ &= \frac{e^c n}{c(n-i)}\left(1-\frac{c}{3+c}+\frac{c'\frac{c}{3+c}}{\mu n(3+c)+c'}\right)+\mathcal{O}(\mu) \\ &= \frac{e^c n}{c(n-i)}\left(1-\frac{c}{3+c}+\mathcal{O}(1/n)\right)+\mathcal{O}(\mu) \\ &= \frac{e^c n}{c(n-i)}\frac{3}{3+c}+\mathcal{O}(\mu). \end{aligned}$$

If we add the expected time to take over each fitness level from Lemma 6 and sum over all fitness levels the upper bound

on the runtime is:

$$\sum_{i=n-n/(4c(1+e^c))}^{n} \left(\frac{e^c n}{c(n-i)} \frac{3}{3+c} + \mathcal{O}(\mu) + \mathcal{O}(\mu \log \mu) \right)$$
$$\leq \sum_{i=0}^{n} \left(\frac{e^c n}{c(n-i)} \frac{3}{3+c} + \mathcal{O}(\mu \log \mu) \right)$$
$$\leq \frac{3e^c n \log n}{c(3+c)} + \mathcal{O}(n\mu \log \mu) \leq \frac{3e^c n \log n}{c(3+c)} (1 + o(1)),$$

where in the last inequality we use $\mu = o(\log n / \log \log n)$ to prove the second statement of the theorem. □

The second statement of the theorem provides an upper bound of $(3/4)en \log n$ for the standard mutation rate $1/n$ (i.e., $c = 1$) and $\mu = o(\log n / \log \log n)$. The upper bound is minimised for $c = \frac{1}{2}\left(\sqrt{13} - 1\right)$. Hence, the best upper bound is delivered for a mutation rate of about $1.3/n$. The resulting leading term of the upper bound is:

$$E[T] \leq \frac{6e^{\frac{1}{2}\left(\sqrt{13}-1\right)} n \log n}{\left(\sqrt{13}-1\right)\left(\frac{1}{2}\left(\sqrt{13}-1\right)+3\right)} \approx 1.97 n \log n.$$

We point out that Theorem 9 holds for any $\mu \geq 3$. Our framework provides a higher upper bound when $\mu = 2$ compared to larger values of μ. The main difference lies in the probability p_r as shown in the following lemma.

Lemma 10. *The transition probabilities p_m, p_r, p_c and p_d for the (2+1) GA, with mutation rate c/n and c constant, are bounded as follows:*

$$p_d \geq \frac{2}{3} \frac{i(n-i)c^2}{n^2(e^c + \mathcal{O}(1/n))}, \qquad p_c \geq \frac{1}{8(e^c + \mathcal{O}(1/n))},$$
$$p_r \leq \frac{5}{24e^c} + \mathcal{O}(1/n), \qquad p_m \geq \frac{c(n-i)}{(e^c + \mathcal{O}(1/n))n}.$$

The upper bound on p_r from Lemma 8 is $1/(8e^c)$, which is smaller than the bound we have just found. This is due to the assumptions in the lemma that there can be only one genotype in the population at a given time which can take over the population in the next iteration. However, when $\mu = 2$, either individual can take over the population in the next iteration. This larger upper bound on p_r for $\mu = 2$ leads to a larger upper bound on the runtime of $E[T] \leq \frac{4}{c+4} \frac{e^c n \log n}{c}(1 + o(1))$ for the (2+1) GA. The calculations are omitted as they are the same as those of the proof of Theorem 9 where $p_r \geq 5/(24e^c) + \mathcal{O}(1/n)$ is used and μ is set to 2.

VI. Lower bound

In the previous section we provided a higher upper bound for the $(2+1)$ GA compared to the $(\mu+1)$ GA with population size greater than 2 and $\mu = o(\log n / \log \log n)$. To rigorously prove that the $(2+1)$ GA is indeed slower, we require a lower bound on the runtime of the algorithm that is higher than the upper bound provided in the previous section for the $(\mu + 1)$ GA ($\mu \geq 3$).

Since providing lower bounds on the runtime is a notoriously hard task, we will follow a strategy previously used by Sudholt [40] and analyse a version of the (μ+1) GA with greedy parent selection and greedy crossover (i.e., Algorithm 2) in the sense that:

Algorithm 2: $(2+1)_S$ GA

1 $P \leftarrow \mu$ individuals, uniformly at random from $\{0,1\}^n$;
2 **repeat**
3 Choose $x, y \in P$ uniformly at random among P^*, the individuals with the current best fitness f^*;
4 $z \leftarrow$ Uniform crossover with probability $1/2$ (x, y);
5 Flip each bit in z with probability c/n;
6 **If** $f(z) = f^*$ and $\max_{w \in P^*}(HD(w,z)) > 2$ **then** $z \leftarrow z \vee \arg\max_{w \in P^*}(HD(w,z))$;
7 $P \leftarrow P \cup \{z\}$;
8 Choose one element from P with lowest fitness and remove it from P, breaking ties at random;
9 **until** *termination condition satisfied*;

1) Parents are selected uniformly at random only among the solutions from the highest fitness level (*greedy selection*).
2) If the offspring has the same fitness as its parents and its Hamming distance to any individual with equal fitness in the population is larger than 2, then the algorithm automatically performs an OR operation between the offspring and the individual with the largest Hamming distance and fitness, breaking ties arbitrarily, and adds the resulting offspring to the population i.e., we pessimistically allow it to skip as many fitness levels as possible (*semi-greedy crossover*).

The greedy selection allows us to ignore the improvements that occur via crossover between solutions from different fitness levels. Thus, the crossover is only beneficial when there are at least two different genotypes in the population at the highest fitness level discovered so far. The difference with the algorithm analysed by Sudholt [40] is that the $(2+1)_S$ GA we consider does not use any diversity mechanism and it does not automatically crossover correctly when the Hamming distance between parents is exactly 2. As a result, there still is a non-zero probability of losing diversity before a successful crossover occurs. The crossover operator of the $(2+1)_S$ GA is less greedy than the one analysed in [40] (i.e., there crossover is automatically successful also when the Hamming distance between the parents is 2). We point out that the upper bounds on the runtime derived in the previous section also hold for the greedy $(2+1)_S$ GA.

The Markov chain structure of Figure 1 is still representative of the states that the algorithm can be in. When there is no diversity in the population, either an improvement via mutation occurs or diversity is introduced into the population by the mutation operator. When diversity is present, both crossover and mutation can reach a higher fitness level while there is also a probability that the population will lose diversity by replicating one of the existing genotypes. With a population size of two the diversity can be lost by creating a copy of either solution and removing the other one from the population during *environmental selection* (i.e., Line 8 in Algorithm 2).

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2745715, IEEE Transactions on Evolutionary Computation

Fig. 2. Average runtime over 1000 independent runs versus problem size n.

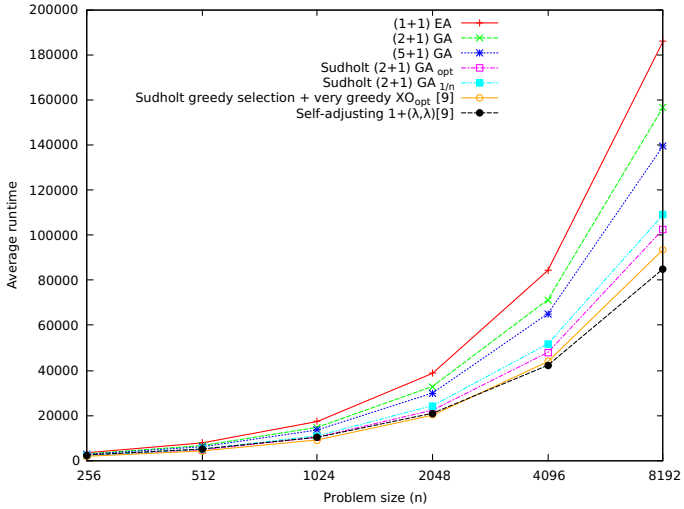
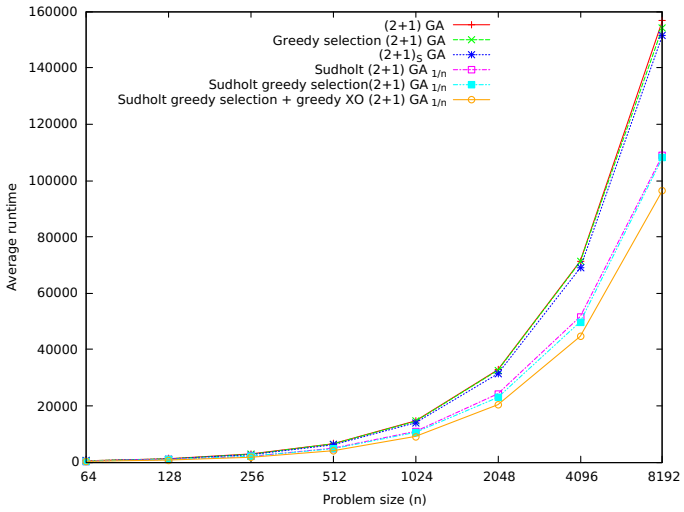


Fig. 3. Comparison between standard selection, greedy selection and greedy selection + greedy crossover GAs. The runtime is averaged over 1000 independent runs.



With population sizes greater than two, the loss of diversity can only occur when the majority genotype (i.e., the genotype with most copies in the population) is replicated. Building upon this we will show that the asymptotic performance of $(2+1)_S$ GA for ONEMAX cannot be better than that of the (μ+1) GAs for $\mu > 2$.

Like in [40] for our analysis we will apply the fitness level method for lower bounds proposed by Sudholt [39]. Due to the greedy crossover and the greedy parent selection used in [40], the population could be represented by the trajectory of a single individual. If an offspring with lower fitness was added to the population, then the greedy parent selection never chose it. If instead, a solution with equally high fitness and different genotype was created, then the algorithm immediately reduced the population to a single individual that is the best possible outcome from crossing over the two genotypes. The main difference between the following analysis and that of [40] is that we want to take into account the possibility that the gained diversity may be lost before crossover exploits it. To this end, when individuals of equal fitness and Hamming distance 2 are created, crossover only exploits this successfully (i.e., goes to the next fitness level) with the conditional probability that crossover is successful before the diversity is lost. Otherwise, the diversity is lost. Only when individuals of Hamming distance larger than 2 are created, we allow crossover to immediately provide the best possible outcome as in [40].

Now, we can state the main result of this section.

Theorem 11. *The expected runtime of the $(2+1)_S$ GA with mutation probability $p = c/n$ for any constant c on* ONEMAX *is no less than:*

$$\frac{3e^c}{c\left(3+\max\limits_{1\le k\le n}\left(\frac{(np)^k}{(k!)^2}\right)\right)} n\log n - \mathcal{O}(n\log\log n).$$

Note that for $c \le 4$, $\max\limits_{1\le k\le n}\left(\frac{(np)^k}{(k!)^2}\right) \le pn = c$. Since $E[T] \ge en\log n$ for $c \ge 3$, for the purpose of finding the mutation rate that minimises the lower bound, we can reduce the statement of the theorem to:

$$\frac{3e^c n\log n}{c(3+c)} - \mathcal{O}(n\log\log n).$$

The theorem provides a lower bound of $(3/4)en\log n - \mathcal{O}(n\log\log n)$ for the standard mutation rate $1/n$ (i.e., c = 1). The lower bound is minimised for $c = \frac{1}{2}\left(\sqrt{13}-1\right)$. Hence, the smallest lower bound is delivered for a mutation rate of about $1.3/n$. The resulting lower bound is:

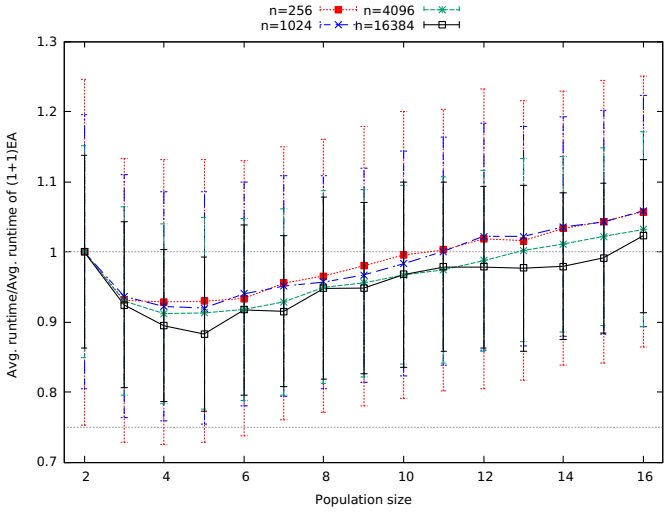
$$\begin{aligned} E[T] \ge & \frac{6e^{\frac{1}{2}\left(\sqrt{13}-1\right)} n\log n}{\left(\sqrt{13}-1\right)\left(\frac{1}{2}\left(\sqrt{13}-1\right)+3\right)} - \mathcal{O}(n\log\log n) \\ \approx & 1.97n\log n - \mathcal{O}(n\log\log n). \end{aligned}$$

Since the lower bound for the $(2+1)_S$ GA matches the upper bound for the (μ+1) GA with $\mu > 2$, the theorem proves that, under greedy selection and semi-greedy crossover, populations of size 2 cannot be faster than larger population sizes up to $\mu = o(\log n/\log\log n)$. In the following section we give experimental evidence that the greedy algorithms are faster than the standard (2+1) GA, thus suggesting that the same conclusions hold also for the standard non-greedy algorithms.

VII. EXPERIMENTS

The theoretical results presented in the previous sections pose some new interesting questions. On one hand, the theory suggests that population sizes greater than 2 benefit the (μ+1) GA for hillclimbing the ONEMAX function. On the other hand, the best runtime bounds are obtained for a mutation rate of approximately $1.3/n$, suggesting that higher mutation rates than the standard $1/n$ rate may improve the performance of the (μ+1) GA. In this section we present the outcome of some experimental investigations to shed further light on these questions. In particular, we will investigate the

Fig. 4. Average runtime gain of the (μ+1) GA versus the (2+1) GA for different population sizes, errorbars show the standard deviation normalised by the average runtime for $\mu = 2$.



effects of the population size and mutation rate on the runtime of the steady-state GA for ONEMAX and compare its runtime against other GAs that have been proved to be faster than mutation-only EAs in the literature.

We start with an overview of the performance of the algorithms. In Fig. 2, we plot the average runtime over 1000 independent runs of the (μ+1) GA with $\mu = 2$ and $\mu = 5$ (with uniform parent selection and standard $1/n$ mutation rate) for exponentially increasing problem sizes and compare it against the fastest standard bit mutation-only EA with static mutation rate (i.e., the (1+1) EA with $1/n$ mutation rate). While the algorithm using $\mu = 5$ outperforms the $\mu = 2$ version, they are both faster than the (1+1) EA already for small problem sizes. We also compare the algorithms against the (2+1) GA investigated by Sudholt [40] where diversity is enforced by the environmental selection always preferring distinct individuals of equal fitness - the same GA variant that was first proposed and analysed in [23]. We run the algorithm both with standard mutation rate $1/n$ and with the optimal mutation rate $(1 + \sqrt{5})/(2n)$. Obviously, when diversity is enforced, the algorithms are faster. Finally, we also compare the algorithms against the (1+(λ,λ)) GA with self-adjusting population sizes and Sudholt's (2+1) GA as they were compared previously in [10]. Note that in [10] (Fig. 8 therein) Sudholt's algorithm was implemented with a very greedy parent selection operator that always prefers distinct individuals on the highest fitness level for reproduction.

In order to decompose the effects of the greedy parent selection, greedy crossover and the use of diversity, we conducted further experiments shown in Figure 3. Here, we see that it is indeed the enforced diversity that creates the fundamental performance difference. Moreover, the results show that the greedy selection/greedy crossover GA is slightly faster than the greedy parent selection GA and that greedy parent selection is slightly faster than standard selection. Overall, the figure suggests that the lower bound presented in Theorem 11 is also valid for the standard (2+1) GA with uniform parent selection (i.e., no greediness). In Figure 3, it can be noted that the performance difference between the GA with greedy crossover and greedy parent selection analysed in [40] and the (2+1) GA with enforced diversity and without greedy crossover is more pronounced than the performance difference between the standard (2+1) GA analysed in Section V and the $(2+1)_S$ GA which was analysed in Section VI. The reason behind the difference in discrepancies is that the $(2+1)_S$ GA does not implement the greedy crossover operator when the Hamming distance is 2. We speculate that cases where the Hamming distance is *just* enough for the crossover to exploit it occur much more frequently than the cases where a larger Hamming distance is present. As a result, the performance of the $(2+1)_S$ GA does not deviate much from the standard algorithm. Table I (see supplementary material) presents the mean and standard deviation of the runtimes of the algorithms depicted in Figure 2 and Figure 3 over 1000 independent runs.

Now we investigate the effects of the population size on the (μ+1) GA. We perform 1000 independent runs of the (μ+1) GA with uniform parent selection and standard mutation rate $1/n$ for increasing population sizes up to $\mu = 16$. In Fig. 4 we present average runtimes divided by the runtime of the $(2+1)$ GA and in Fig. 5 normalised against the runtime of the (1+1) EA. In both figures, we see that the runtime improves for μ larger than 2 and after reaching its lowest value increases again with the population size. It is not clear whether there is a constant optimal static value for μ around 4 or 5. The experiments, however, do not rule out the possibility that the optimal static population size increases slowly with the problem size (i.e., $\mu = 3$ for $n = 256$, $\mu = 4$ for $n = 4096$ and $\mu = 5$ for $n = 16384$). On the other hand, we clearly see that as the problem size increases we get a larger improvement on the runtime. This indicates that the harder is the problem, more useful are the populations. In particular, in Figure 5 we see that the theoretical asymptotic gain of 25% with respect to the runtime of the (1+1) EA is approached more and more closely as n increases. For the considered problem sizes, the (μ+1) GA is faster than the (1+1) EA for all tested values of μ. However, to see the runtime improvement of the (μ+1) GA against the (2+1) GA for $\mu > 15$ the experiments (Fig. 4) suggest that greater problem sizes would need to be used.

Finally, we investigate the effect of the mutation rate on the runtime. Based on our previous experiments we set the population size to the best seen value of $\mu = 5$ and perform 1000 independent runs for each c value ranging from 0.9 to 1.9. In Figure 6, we see that even though the mutation rate $c \approx 1.3$ minimises the upper bound we proved on the runtime, setting a larger mutation rate of 1.6 further decreases the runtime.

VIII. CONCLUSION

The question of whether genetic algorithms can hillclimb faster than mutation-only algorithms is a long standing one. On one hand, in his pioneering book, Rechenberg had given preliminary experimental evidence that crossover may speed up the runtime of population based EAs for generalised ONEMAX [34]. On the other hand, further experiments suggested that genetic algorithms were slower hillclimbers than

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TEVC.2017.2745715, IEEE Transactions on Evolutionary Computation

Fig. 5. Average runtime gain of the (μ+1) GA versus the (1+1) EA for different population sizes, errorbars show the standard deviation normalised by the average runtime of the (1+1) EA.

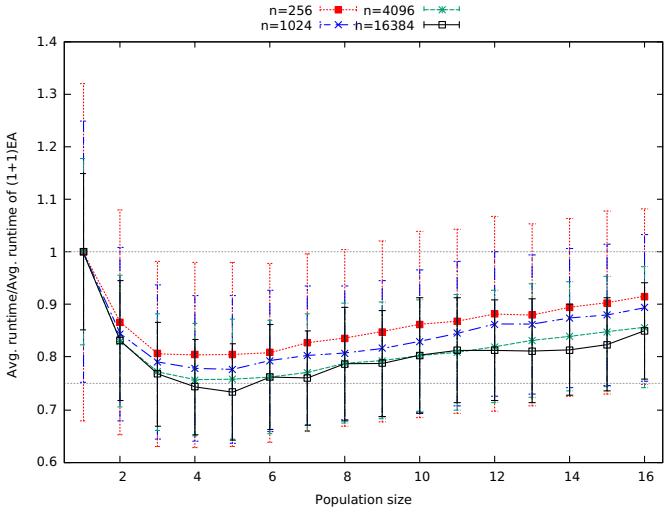
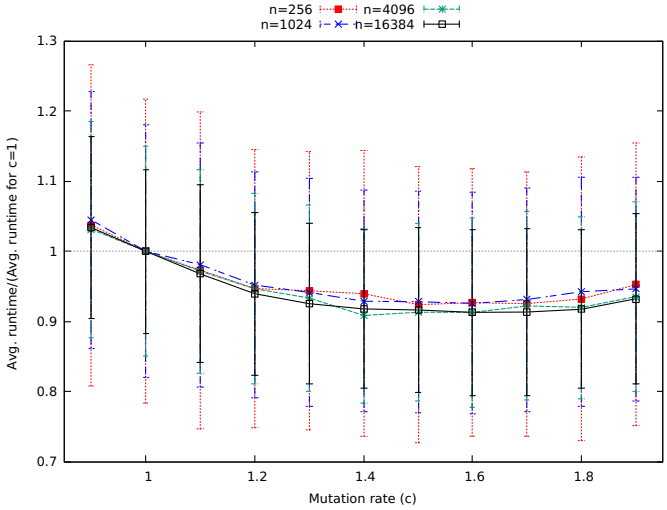


Fig. 6. Average runtime gain of the (5+1) GA for various mutation rates versus the standard $1/n$ mutation rate, errorbars show the standard deviation normalised by the average runtime for $1/n$ mutation rate.



the (1+1) EA [21], [29]. In recent years it has been rigorously shown that crossover and mutation can outperform algorithms using only mutation. Firstly, a new theory-driven GA called (1+(λ,λ)) GA has been shown to be asymptotically faster for hillclimbing the ONEMAX function than any unbiased mutation-only EA [10]. Secondly, it has been shown how standard (μ+λ) GAs are twice as fast as their standard bit mutation-only counterparts for ONEMAX as long as diversity is enforced through environmental selection [40].

In this paper we have rigorously proven that standard steady-state GAs with $\mu \geq 3$ and $\mu = o(\log n / \log \log n)$ are at least 25% faster than all unbiased standard bit mutation-based EAs with static mutation rate for ONEMAX even if no diversity is enforced. The Markov Chain framework we used to achieve the upper bounds on the runtimes should be general enough to allow future analyses of more complicated GAs, for instance with greater offspring population sizes or more sophisticated crossover operators. A limitation of the approach is that it applies to classes of problems that have plateaus of equal fitness. Hence, for functions where each genotype has a different fitness value our approach would not apply. An open question is whether the limitation is inherent to our framework or whether it is crossover that would not help steady-state EAs at all on such fitness landscapes.

Our results also explain that populations are useful not only in the exploration phase of the optimization, but also to improve exploitation during the hillclimbing phases. In particular, larger population sizes increase the probability of creating and maintaining diversity in the population. This diversity can then be exploited by the crossover operator. Recent results had already shown how the interplay between mutation and crossover may allow the emergence of diversity, which in turn allows to escape plateaus of local optima more efficiently compared to relying on mutation alone [5]. Our work sheds further light on the picture by showing that populations, crossover and mutation together, not only may escape optima more efficiently, but may be more effective also in the exploitation phase.

Another additional insight gained from the analysis is that the standard mutation rate $1/n$ may not be optimal for the (μ+1) GA on ONEMAX. This result is also in line with, and nicely complements, other recent findings concerning steady state GAs. For escaping plateaus of local optima it has been recently shown that increasing the mutation rate above the standard $1/n$ rate leads to smaller upper bounds on escaping times [5]. However, when jumping large low-fitness valleys, mutation rates of about $2.6/n$ seem to be optimal static rates (see the experiment section in [3], [4]). For ONEMAX lower mutation rates seem to be optimal static rates, but still considerably larger than the standard $1/n$ rate.

New interesting questions for further work have spawned. Concerning population sizes an open problem is to rigorously prove whether the optimal size grows with the problem size and at what rate. Also determining the optimal mutation rate remains an open problem. While our theoretical analysis delivers the best upper bound on the runtime with a mutation rate of about $1.3/n$, experiments suggest a larger optimal mutation rate. Interestingly, this experimental rate is very similar to the optimal mutation rate (i.e., approximately $1.618/n$) of the (μ+1) GA with enforced diversity proven in [40]. The benefits of higher than standard mutation rates in elitist algorithms is a topic that is gaining increasing interest [31], [2], [15].

Further improvements may be achieved by dynamically adapting the population size and mutation rate during the run. Advantages, in this sense, have been shown for the (1+(λ,λ)) GA by adapting the population size [7] and for single individual algorithms by adapting the mutation rate [13], [27]. Generalising the results to larger classes of hillclimbing problems is intriguing. In particular, proving whether speed ups of the (μ+1) GA compared to the (1+1) EA are also achieved for royal road functions would give a definitive answer to a long standing question [29]. Analyses for larger problem classes such as linear functions and classical combi-

natorial optimisation problems would lead to further insights. Yet another natural question is how the (μ+1) GA hillclimbing capabilities compare to (μ+λ) GAs and generational GAs.

Acknowledgement

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

REFERENCES

[1] T. Bäck, *Evolutionary Algorithms in Theory and Practice*. Oxford University Press, 1996.

[2] D. Corus, P. S. Oliveto, and D. Yazdani, "On the runtime analysis of the opt-ia artificial immune system," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017)*. ACM, 2017, pp. 83–90.

[3] D.-C. 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 or Reinforced Diversity," *ArXiv*, Aug. 2016.

[4] ——, "Escaping local optima using crossover with emergent diversity," *IEEE Transactions on Evolutionary Computation*, pp. –, 2017, in Press.

[5] D. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, and A. M. Sutton, "Emergence of diversity and its benefits for crossover in genetic algorithms," in *International Conference on Parallel Problem Solving from Nature (PPSN XIV)*, 2016, pp. 890–900.

[6] D. Dang, T. Friedrich, T. Kötzing, M. S. Krejca, P. K. Lehre, P. S. Oliveto, D. Sudholt, and A. M. Sutton, "Escaping local optima with diversity mechanisms and crossover," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2016)*. ACM, 2016, pp. 645–652.

[7] B. Doerr and C. Doerr, "Optimal parameter choices through self-adjustment: Applying the 1/5-th rule in discrete settings," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015)*. ACM, 2015, pp. 1335–1342.

[8] ——, "A tight runtime analysis of the (1+(λ, λ)) genetic algorithm on onemax," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2015)*. ACM, 2015, pp. 1423–1430.

[9] B. Doerr, C. Doerr, and F. Ebel, "From black-box complexity to designing new genetic algorithms," *Theoretical Computer Science*, vol. 567, pp. 87 – 104, 2015.

[10] ——, "From black-box complexity to designing new genetic algorithms," *Theoretical Computer Science*, vol. 567, pp. 87–104, 2015.

[11] B. Doerr, C. Doerr, and T. Kötzing, "The right mutation strength for multi-valued decision variables," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2016)*. ACM, 2016, pp. 1115–1122.

[12] B. Doerr, C. Doerr, R. Spöhel, and T. Henning, "Playing mastermind with many colors," *Journal of the ACM*, vol. 63, no. 5, pp. 42:1–42:23, 2016.

[13] B. Doerr, C. Doerr, and J. Yang, "k-bit mutation with self-adjusting k outperforms standard bit mutation," in *International Conference on Parallel Problem Solving from Nature (PPSN XIV)*. Springer, 2016, pp. 824–834.

[14] B. Doerr, E. Happ, and C. Klein, "Crossover can provably be useful in evolutionary computation," *Theoretical Computer Science*, vol. 425, pp. 17–33, 2012.

[15] B. Doerr, H. P. Le, R. Makhmara, and T. D. Nguyen, "Fast genetic algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017)*. ACM, 2017, pp. 777–784.

[16] B. Doerr and C. Winzen, "Playing mastermind with constant-size memory," *Theory of Computing Systems*, vol. 55, no. 4, pp. 658–684, 2014.

[17] A. E. Eiben and J. E. Smith, *Introduction to evolutionary computing*. Springer, 2003.

[18] T. Friedrich, P. S. Oliveto, D. Sudholt, and C. Witt, "Analysis of diversity-preserving mechanisms for global exploration," *Evolutionary Computation*, vol. 17, no. 4, pp. 455–476, 2009.

[19] D. E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*. Addison-Wesley, 1989.

[20] J. H. Holland, *Adaptation in Natural and Artificial Systems*. University Michigan Press, 1975.

[21] T. Jansen and I. Wegener, "Real royal road functions — where crossover provably is essential," *Discrete Applied Mathematics*, vol. 149, no. 1-3, pp. 111–125, 2005.

[22] T. Jansen, *Analyzing evolutionary algorithms: The computer science perspective*. Springer, 2013.

[23] T. Jansen and I. Wegener, "The analysis of evolutionary algorithms-a proof that crossover really can help," *Algorithmica*, vol. 34, no. 1, pp. 47–66, 2002.

[24] T. Kötzing, D. Sudholt, and M. Theile, "How crossover helps in Pseudo-Boolean optimization," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM Press, 2011, pp. 989–996.

[25] P. K. Lehre and C. Witt, "Black-box search by unbiased variation," *Algorithmica*, vol. 64, no. 4, pp. 623–642, 2012.

[26] P. K. Lehre and X. Yao, "Crossover can be constructive when computing unique input–output sequences," *Soft Computing*, vol. 15, no. 9, pp. 1675–1687, 2011.

[27] A. Lissovoi, P. S. Oliveto, and J. A. Warwicker, "On the runtime analysis of generalised selection hyper-heuristics for pseudo-boolean optimisation," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2017)*. ACM, 2017, pp. 849–856.

[28] M. Mitchell, J. Holland, and S. Forrest, "Relative building-block fitness and the building block hypothesis," in *Foundations of Genetic Algorithms (FOGA '93)*, vol. 2, 1993, pp. 109–126.

[29] ——, "When will a genetic algorithm outperform hill climbing?" in *Neural Information Processing Systems (NIPS 6)*. Morgan Kaufmann, 1994, pp. 51–58.

[30] F. Neumann, P. S. Oliveto, G. Rudolph, and D. Sudholt, "On the effectiveness of crossover for migration in parallel evolutionary algorithms," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2011)*. ACM Press, 2011, pp. 1587–1594.

[31] P. S. Oliveto, P. K. Lehre, and F. Neumann, "Theoretical analysis of rank-based mutation-combining exploration and exploitation," in *IEEE Congress on Evolutionary Computation (CEC 2009)*. ACM, 2017, pp. 1455–1462.

[32] P. S. Oliveto and X. Yao, "Runtime analysis of evolutionary algorithms for discrete optimization," in *Theory of Randomized Search Heuristics: Foundations and Recent Developments*, B. Doerr and A. Auger, Eds. World Scientific, 2011, pp. 21–52.

[33] C. Qian, Y. Yu, and Z. Zhou, "An analysis on recombination in multi-objective evolutionary optimization," *Artificial Intelligence*, vol. 204, pp. 99 – 119, 2013.

[34] I. Rechenberg, *Evolutionsstrategie: Optimierung technischer Systeme nach Prinzipien der biologishen Evolution*. Friedrich Frommann Verlag, 1973.

[35] J. E. Rowe, "Genetic algorithms," in *Handbook of computational Intelligence*, W. Pedrycz and J. Kacprzyk, Eds. Springer, 2011, pp. 825–844.

[36] J. Sarma and K. D. Jong, "Generation gap methods," in *Handbook of evolutionary computation*, T. Back, D. B. Fogel, and Z. Michalewicz, Eds. IOP Publishing Ltd., 1997, ch. C 2.7.

[37] T. Storch and I. Wegener, "Real royal road functions for constant population size," *Theoretical Computer Science*, vol. 320, no. 1, pp. 123–134, 2004.

[38] D. Sudholt, "Crossover is provably essential for the ising model on trees," in *Proceedings of the Genetic and Evolutionary Computation Conference (GECCO 2005)*. New York, New York, USA: ACM Press, 2005, pp. 1161–1167.

[39] ——, "A new method for lower bounds on the running time of evolutionary algorithms," *IEEE Transactions on Evolutionary Computation*, vol. 17, no. 3, pp. 418–435, 2013.

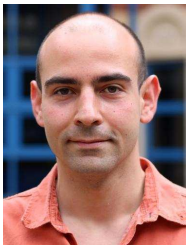
[40] ——, "How crossover speeds up building-block assembly in genetic algorithms," *Evolutionary Computation*, vol. 25, no. 2, pp. 237–274, 2017.

[41] C. Witt, "Runtime analysis of the (μ+1) ea on simple pseudo-boolean functions," *Evolutionary Computation*, vol. 14, no. 1, pp. 65–86, 2006.

[42] ——, "Tight bounds on the optimization time of a randomized search heuristic on linear functions," *Combinatorics, Probability & Computing*, vol. 22, no. 2, pp. 294–318, 2013.

Pietro S. Oliveto is a Senior Lecturer and an EPSRC Early Career Fellow at the University of Sheffield, UK. He received the Laurea degree in computer science from the University of Catania, Italy in 2005 and the PhD degree from the University of Birmingham, UK in 2009. He has been EPSRC PhD+ Fellow (2009-2010) and EPSRC Postdoctoral Fellow (2010-2013) at Birmingham and Vice-Chancellor's Fellow at Sheffield (2013-2016).

His main research interest is the performance analysis of bio-inspired computation techniques including evolutionary algorithms, genetic programming, artificial immune systems and hyperheuristics. He has won best paper awards at GECCO 2008, ICARIS 2011 and GECCO 2014. He is part of the Steering Committee of the annual workshop on Theory of Randomized Search Heuristics (ThRaSH), Associate Editor of the IEEE Transactions on Evolutionary Computation, Chair of the IEEE CIS Task Force on Theoretical Foundations of Bio-inspired Computation and member of the EPSRC Peer Review College.



Dogan Corus is a Research Associate at the University of Sheffield, UK.

He received his B.S. and M.S. in industrial engineering from Koc University, Istanbul, Turkey, and his Ph. D. in computer science from the University of Nottingham, UK.

His main research interest is the runtime analysis of bio-inspired algorithms, in particular, population based algorithms and genetic algorithms.