Algorithmica (2020) 82:3676–3706
https://doi.org/10.1007/s00453-020-00743-1

On the Benefits of Populations for the Exploitation Speed of Standard Steady-State Genetic Algorithms

Dogan Corus[1] · **Pietro S. Oliveto**[1]

Received: 21 October 2019 / Accepted: 23 June 2020 / Published online: 15 July 2020

Abstract

It is generally accepted that populations are useful for the global exploration of multi-modal optimisation problems. Indeed, several theoretical results are available showing such advantages over single-trajectory search heuristics. In this paper we provide evidence that evolving populations via crossover and mutation may also benefit the optimisation time for hillclimbing unimodal functions. In particular, we prove bounds on the expected runtime of the standard $(\mu + 1)$ GA for OneMax that are lower than its unary black box complexity and decrease in the leading constant with the population size up to $\mu = o\left(\sqrt{\log n}\right)$. Our analysis suggests that the optimal mutation strategy is to flip two bits most of the time. To achieve the results we provide two interesting contributions to the theory of randomised search heuristics: (1) A novel application of drift analysis which compares absorption times of different Markov chains without defining an explicit potential function. (2) The inversion of fundamental matrices to calculate the absorption times of the Markov chains. The latter strategy was previously proposed in the literature but to the best of our knowledge this is the first time is has been used to show non-trivial bounds on expected runtimes.

Keywords Genetic algorithms · Crossover · Recombination · Population-based algorithms · Theory · Runtime analysis

An extended abstract of this paper was published in [3].

✉ Dogan Corus
d.corus@sheffield.ac.uk

Pietro S. Oliveto
p.oliveto@sheffield.ac.uk

[1] The University of Sheffield, Sheffield, UK

1 Introduction

Populations in evolutionary and genetic algorithms are considered crucial for the effective global optimisation of multi-modal problems. For this to be the case, the population should be sufficiently diverse such that it can explore multiple regions of the search space at the same time [10]. Also, if the population has sufficient diversity, then it considerably enhances the effectiveness of crossover for escaping from local optima. Indeed the first proof that crossover can considerably improve the performance of GAs relied on either enforcing diversity by not allowing genotypic duplicates or by using unrealistically small crossover rates for the so-called Jump function [14]. Since then, it has been shown several times that crossover is useful to GAs using the same, or similar, diversity enhancing mechanisms for a range of optimisation problems including shortest path problems [5], vertex cover [23], colouring problems inspired by the Ising model [32] and computing input output sequences in finite state machines [17].

These examples provide considerable evidence that, by enforcing the necessary diversity, crossover makes GAs effective and often superior to applying mutation alone. However, rarely it has been proven that the diversity mechanisms are actually necessary for GAs, or to what extent they are beneficial to outperform their mutation-only counterparts rather than being applied to simplify the analysis. Recently, some light has been shed on the power of standard genetic algorithms without diversity over the same algorithms using mutation alone. Dang et al. showed that the plain $(\mu + 1)$ GA is at least a linear factor faster than its $(\mu + 1)$ EA counterpart at escaping the local optimum of Jump [4]. Sutton showed that the same algorithm with crossover if run sufficiently many times is a fixed parameter tractable (FPT) algorithm for the closest string problem while without crossover it is not [33]. Lengler provided an example of a class of unimodal functions to highlight the robustness of the crossover based version with respect to the mutation rate compared to the mutation-only version i.e., the $(\mu + 1)$ GA is efficient for any mutation rate c/n, c a constant, while the $(\mu + 1)$ EA requires exponential time as soon as approx. $c > 2.13$ [19]. In all three examples the population size has to be large enough for the results to hold, thus providing evidence of the importance of populations in combination with crossover. A follow-up work by Lengler and Zou also established that for some monotone functions increasing the population size can result in superpolynomial runtimes when crossover is not implemented [20].

Recombination has also been shown to be very helpful at exploitation if the necessary diversity is enforced through some mechanism. In the $(1+(\lambda, \lambda))$ GA such diversity is achieved through large mutation rates. The algorithm can optimise the well-known OneMax function in $o(n \log n)$ expected time with static offspring population sizes λ [7], and in linear time with self-adaptive values of λ [6]. Although using a recombination operator, the algorithm is still basically a single-trajectory one (i.e., there is no population). More realistic steady-state GAs that actually create offspring by recombining parents have also been analysed for OneMax. Sudholt showed that $(\mu + \lambda)$ GAs are twice as fast as their mutation-only

version (i.e., no recombination) for OneMax if diversity is enforced artificially i.e., genotype duplicates are preferred for deletion [31]. He proved a runtime of $(e/2)n \ln n + O(n)$ versus the $en \ln n + O(n)$ function evaluations required by any standard bit mutation-only evolutionary algorithm for OneMax and any other linear function [30, 35]. If offspring are identical to their parents it is not necessary to evaluate the quality of their solution. When the unnecessary queries are avoided, the expected runtime of the GA using artificial diversity from [31] is bounded above by $(1 + o(1))0.850953n \ln n$ [28]. Hence, it is faster than any unary (i.e., mutation-only) unbiased[1] black-box search heuristic [16].

On one hand, the enforced artificiality in the last two results considerably simplifies the analysis. On the other hand, the power of evolving populations for effective optimisation cannot be appreciated. Since the required diversity to make crossover effective is artificially enforced, the optimal population size is 2 and larger populations provide no benefits. Corus and Oliveto showed that the standard $(\mu + 1)$ GA without diversity is still faster for OneMax than mutation-only ones by proving an upper bound on the runtime of $(3/4)en \ln n + O(n)$ for any $3 < \mu < o(\log n / \log \log n)$ [2]. A result of enforcing the diversity in [31] was that the best GA for the problem only used a population of size 2. However, even though this artificiality was removed in [2], a population of size 3 was sufficient to get the best upper bound on the runtime achievable with their analysis. Overall, their analysis does not indicate any tangible benefit towards using a population larger than $\mu = 3$. Thus, rigorously showing that populations are beneficial for GAs in the exploitation phase has proved to be a non-trivial task.

In this paper we provide a more precise analysis of the behaviour of the population of the $(\mu + 1)$ GA for OneMax. We prove that the standard $(\mu + 1)$ GA with $\mu = o(\sqrt{\log n})$ is at least 60% faster than the same algorithm using only mutation. We also prove that the GA is faster than any unary unbiased black-box search heuristic if offspring with identical genotypes to their parents are not evaluated. More importantly, our upper bounds on the expected runtime decrease with the population size up to $\mu = o(\sqrt{\log n})$, thus providing for the first time a natural example where *populations* evolved via recombination and mutation optimise faster than any unary unbiased search heuristic. The mutation rate that minimises our upper bounds is approximately $1.4/n$. With such rates all population sizes $\mu \geq 5$ have an expected runtime that is smaller than $1.675n \ln n(1 + o(1))$ in the standard case when all offspring are evaluated. A recent analysis that proves that the expected runtime of the (2+1) GA is at least $2.18417n \ln n + O(n)$ for any mutation rate c/n, for any $c < 1.422$ [27] combined with our previously proven upper bounds for $\mu \geq 3$ [2], have allowed the first rigorous proof that any population size greater than $\mu = 2$ makes the $(\mu + 1)$ GA faster than the same algorithm using only 2 individuals (i.e., populations are provably beneficial in the exploitation phase—for hillclimbing OneMax). In this paper we provide guarantees of larger speed-ups. These guarantees increase with the population size μ.

[1] The probability of a bit being flipped by an unbiased operator is the same for each bit-position and bit value.

2 Problem Definition and Our Results

2.1 The Genetic Algorithm

The $(\mu + 1)$ GA is a standard steady-state GA which samples a single new solution at every generation [9, 29]. It keeps a population of the μ best solutions sampled so far and at every iteration selects two solutions from the current population uniformly at random with replacement as the *parents*. The recombination operator then picks building blocks from the parents to create the *offspring* solution. For the case of pseudo-Boolean functions $f : \{0,1\}^n \to \mathbb{R}$, the most frequently used recombination operator is *uniform crossover* which picks the value of each bit position $i \in [n]$ from one parent or the other uniformly at random (i.e., from each parent with probability 1/2) [9]. Then, an unbiased unary variation operator, which is called *the mutation operator*, is applied to the offspring solution before it is added to the population. The most common mutation operator is standard bit mutation which independently flips each bit of the offspring solution with some probability c/n [35]. Finally, before moving to the next iteration, one of the solutions with the worst fitness value is removed from the population. For the case of maximisation the $(\mu + 1)$ GA is defined in Algorithm 1. The runtime of Algorithm 1 is the number of function evaluations until a solution which maximises the function f is sampled for the first time. If every offspring is evaluated, then the runtime is equal to the value of the variable t in Alg. 1 when the optimal solution is sampled. However, if the fitness of offspring which are identical to their parents is not evaluated, then the runtime is smaller than t. We will first analyze the former scheme and then adapt the result to the latter.

Algorithm 1: $(\mu+1)$ GA [9,28]

```
1  P_1 ← μ individuals, uniformly at random from {0,1}^n;
2  t ← μ;
3  repeat
4      Select x, y ∈ P_t uniformly at random with replacement ;
5      z ← uniform crossover(x, y);
6      z ← mutate(z);
7      P_{t+1} ← P_t ∪ {z};
8      Remove the element with lowest fitness from P_{t+1}, breaking ties at
         random;
9      t ← t + 1;
10 until optimum is found;
```

2.2 The Optimisation Problem

Given an unknown bitstring $z \in \{0,1\}^n$, $OneMax_z(x) := |\{i \in [n] \mid z_i = x_i\}|$ returns the number of bits on which a candidate solution $x \in \{0,1\}^n$ matches z [35]. W.l.o.g. we will assume that the target string z of the $OneMax_z$ function to be identified is the bitstring of all one-bits since all the operators in the $(\mu + 1)$ GA are invariant to the bit-value (have no bias towards 0s or 1s).

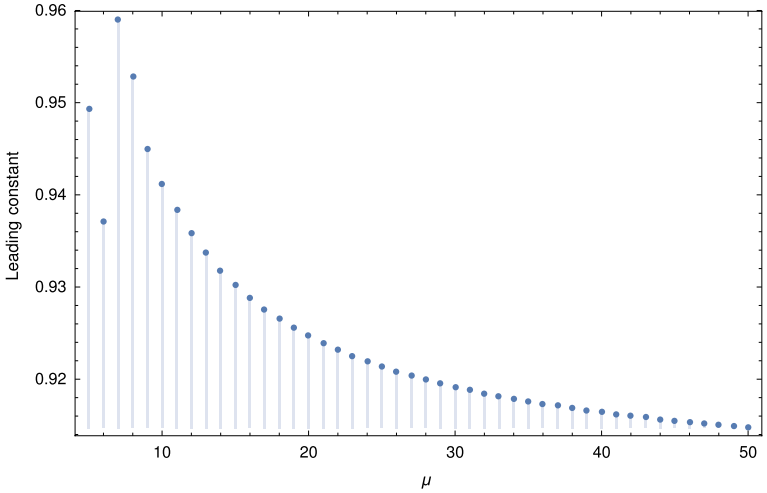

Fig. 1 The leading constant for the $(\mu + 1)$ GA which does not evaluate the fitness of copies versus the population size with $p_0 = \Omega(1)$, $p_1 = \Omega(1)$, $p_2 \approx 1$ and $p_0 + p_1 + p_2 = 1$. The best leading constant achievable by any unary unbiased algorithm is 1. Note that the vertical axis starts at 0.916

2.3 Our Results

In this paper we prove the following results.

Informal *The expected runtime $E[T]$ for the $(\mu + 1)$* GA *with unbiased mutations and population size $\mu = o(\sqrt{\log n})$ to optimise the* OneMax *function is*

1. $E[T] \leq (1 + o(1))n \ln n \cdot \gamma_1(\mu, p_0, p_1, p_2)$, *if offspring identical to their parents are not evaluated and p_0, p_1, and p_2 are respectively the probabilities that zero, one or two bits are flipped*, [Theorem 1, Sect. 3]
2. $E[T] \leq (1 + o(1))n \ln n \cdot \gamma_2(\mu, c)$, *if the quality of each offspring is evaluated and standard bit mutation with rate c/n, $c \in \Theta(1)$, is used*, [Corollary 1, Sect. 3]

where γ_1 and γ_2 are decreasing functions of the population size μ.

The above two statements are very general as they provide upper bounds on the expected runtime of the $(\mu + 1)$ GA for each value of the population size up to $\mu = o(\sqrt{\log n})$ and any unbiased mutation operator. The leading constants γ_1 and γ_2 in Statements 1 and 2 are plotted respectively in Fig. 1 and in Fig. 3 for different population sizes using the p_0, p_1, p_2 and c values which minimise the upper bounds. The result is significant particularly for the following three reasons (in order of increasing importance).

(1) The first statement shows how the genetic algorithm outperforms any unbiased mutation-only heuristic since the best expected runtime achievable by any algorithm belonging to such class is at least $n \ln n - cn \pm o(n)$ [8]. The leading constants in the expected runtime for different population sizes using the p_0, p_1 and p_2 values

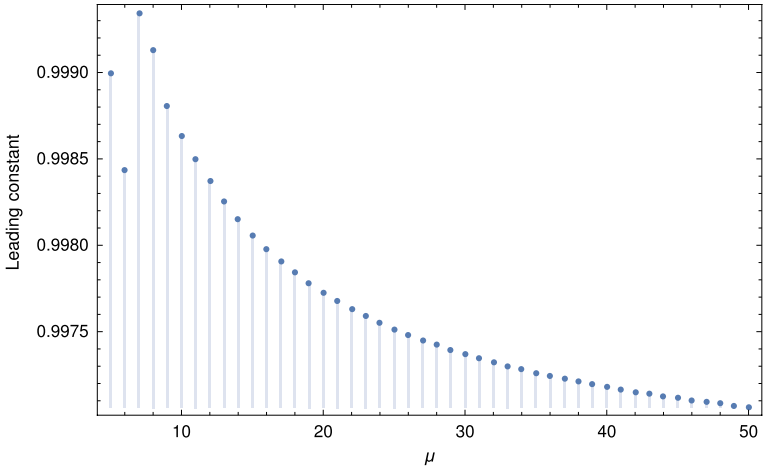

Fig. 2 The leading constant when SBM is used without evaluating offspring duplicates versus the population size. The best leading constant achievable by any unary unbiased algorithm is 1. Note that the vertical axis starts at 0.9966

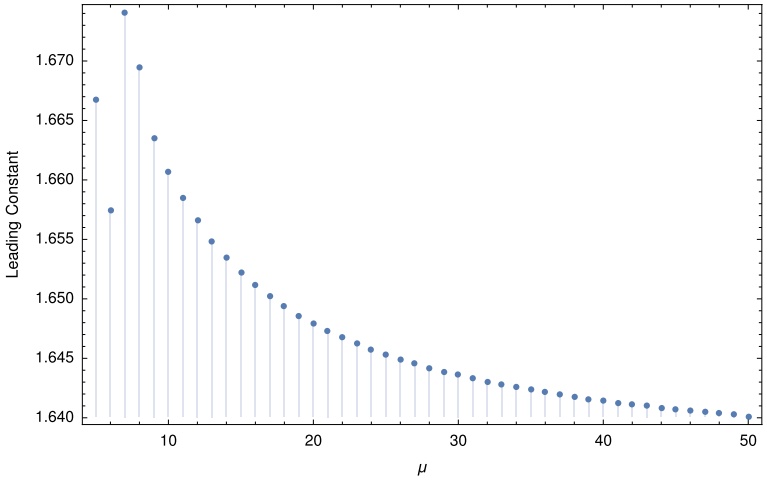

Fig. 3 The leading constant when SBM is used and offspring duplicates are evaluated versus the population size. For each μ value, the corresponding mutation rate minimising the upper bound is used. The best mutation-only variant has a the leading constant of $e \approx 2.71$. Note that the vertical axis starts at 1.639

which minimise the upper bounds (i.e., local mutations) are plotted in Fig. 1 while Fig. 2 shows the leading constants for the standard bit mutation rates c which minimise them. We have assumed $p_2 = 1$ to obtain the leading constants in Fig. 1 since the leading constants in the upper bounds improve as p_2 approaches 1 and we can pick p_2 to be arbitrarily close to 1 (Figs. 3, 4). For the SBM variant, the mutation

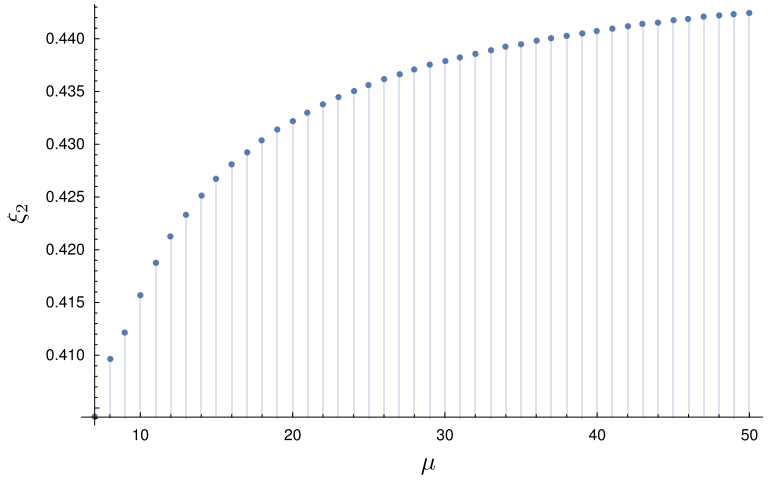

Fig. 4 The value of ξ_2 with respect to the population size $\mu \geq 7$

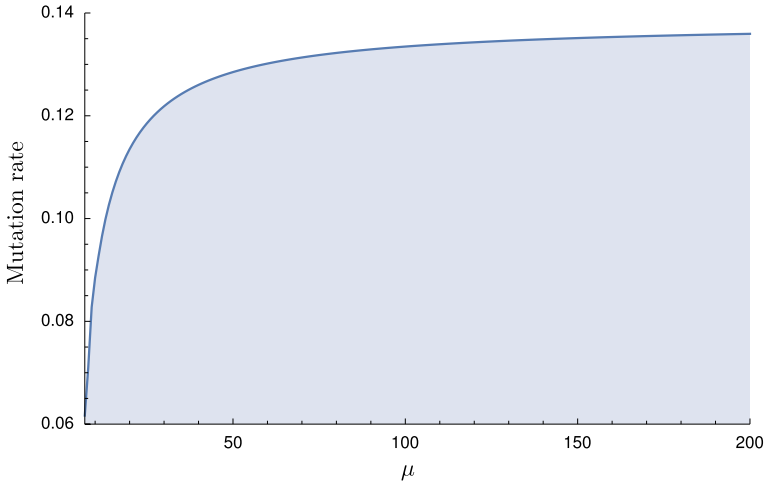

Fig. 5 The mutation rates minimising the upper bounds when SBM is used and the offspring duplicates are not evaluated versus the population size. Note that the x-axis starts at a population size of $\mu = 7$

rates that minimise the upper bounds when the duplicates are not evaluated are plotted for each μ in Fig. 5. It can be appreciated how in both cases all unbiased mutation-only heuristics are outperformed.

Given that the best expected runtime achievable by any search heuristic using only standard bit mutation is $(1 - o(1))en \ln n$ [30, 35], the second statement shows how by adding recombination a speed-up of at least 60% is achieved for the OneMax problem for any population size up to $\mu = o(\sqrt{\log n})$. The leading constants, in this case where offspring are always evaluated, are provided in Fig. 3 and the mutation rates that minimise the upper bound are plotted in Fig. 6.

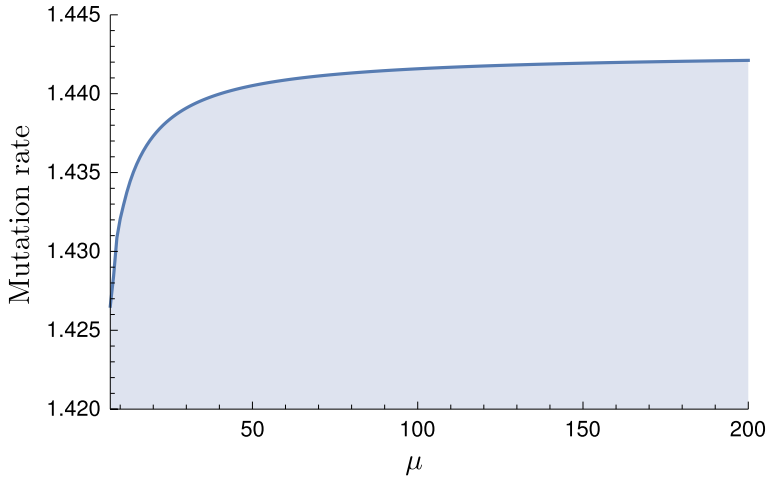

Fig. 6 The mutation rates minimising the upper bounds when SBM is used and the offspring duplicates are evaluated versus the population size. Note that the x-axis starts at a population size of $\mu = 7$

(2) Very few results are available proving constants in the leading terms of the expected runtime for randomised algorithms due to the considerable technical difficulties in deriving them. Exceptions exist such as the analyses of [8, 35] without which our comparative results would not have been achievable. While such precise results are gaining increasing importance in the theoretical computer science community, the available ones are related to more simple algorithms. This is the first time similar results are achieved concerning a much more complicated to analyse standard genetic algorithm using realistic population sizes and recombination.

(3) The preciseness of the analysis allows for the first time an appreciation of the surprising importance of the population for optimising unimodal functions[2] as our upper bounds on the expected runtime decrease as the population size increases. In particular as the problem size increases, so does the optimal size of the population[3] (the best known runtime available for the $(\mu + 1)$ GA was of $(1 + o(1))(3/4)en \ln n$ independent of the population size as long as it is greater than $\mu = 3$ i.e., there were no evident advantages in using a larger population [2]). This result is in contrast to all previous analyses of simplified evolutionary algorithms for unimodal functions where the algorithmic simplifications, made for the purpose of making the analysis more accessible, caused the use of populations to be either ineffective or detrimental [28, 31, 34]. In particular, since a (2+1) GA has an expected runtime of at least $2.18417n \ln n$ for any mutation rate $c < 1.422$ [27], any population size greater than 2 is provably faster (unless higher mutation rates turn out to be beneficial to the (2+1) GA which is unlikely because the optimal mutation rate for the algorithm within the range is approximately $c = 1.2122$). Our upper bound of $\mu = o(\sqrt{\log n})$

[2] Populations are traditionally thought to be useful for solving multi-modal problems.

[3] The population size that minimises the upper bound we obtain.

is very close to the *at most logarithmic* population sizes typically recommended for monotone functions to achieve asymptotically optimal runtimes [1, 34]. We conjecture that the optimal population size is $\Theta\big((\log n)^{1-\epsilon}\big)$ for any constant $\epsilon > 0$, which cannot be proven with our mathematical methods for technical reasons.

2.4 Proof Strategy

Our aim is to provide a precise analysis of the expected runtime of the $(\mu + 1)$ GA for optimising OneMax with arbitrary problem size n. Deriving the exact transition probabilities of the algorithm from all possible configurations of its population to all others is prohibitive. We will instead devise a set of n Markov chains, one for each improvement the algorithm has to make pessimistically to reach the global optimum, which will be easier to analyse. Then we will prove that the Markov chains are slower to reach their absorbing state than the $(\mu + 1)$ GA is in finding the corresponding improvement.

In essence, our proof strategy is: (1) to identify suitable Markov chains, (2) to prove that the absorbing times of the Markov chains are larger than the expected improving times of the actual algorithm and (3) to bound the absorbing times of each Markov chain.

In particular, concerning point (2) we will first define a potential function which monotonically increases with the number of copies of the genotype with most duplicates in the population and then bound the expected change in the potential function at every iteration (i.e., *the drift*) from below. Using the maximum value of the potential function and the minimum drift, we will bound the expected time until the potential function value drops to its minimum value for the first time. This part of the analysis is a novel application of drift analysis techniques [18, 36]. In particular, rather than using an explicit distance function as traditionally occurs, we define the potential function to be equal to the conditional expected absorption time of the corresponding states of each Markov chain.

Concerning point (3) of our proof strategy, we will calculate the absorbing times of the Markov chains M^j by identifying their fundamental matrices. This requires the inversion of tridiagonal matrices. Similar matrix manipulation strategies to bound the runtime of evolutionary algorithms have been previously suggested in the literature [11, 26]. However, all previous applications of the approach proved results that could be trivially achieved via simpler standard methods such as the artificial fitness levels method and random walks [13, 15]. To the best of our knowledge, this is the first time that the power of this long abandoned approach has finally been shown by proving non-trivial bounds on the expected runtime.

3 Main Result Statement

Our main result is the following theorem. The transition probabilities $p_{i,k}$ for $(i, k) \in [m]^2$ and $m := \lceil \mu/2 \rceil$ are defined in Definition 1 (Sect. 4.1) and are depicted in Fig. 9.

Theorem 1 *The expected runtime $E[T]$ for the $(\mu+1)$ GA with $\mu = o(\sqrt{\log n})$ using the uniform crossover and an unbiased mutation operator mutate(x) that flips i bits with probability p_i with $p_0 \in \Omega(1)$ and $p_1 \in \Omega(1)$ to optimise the* OneMax *function is*:

1. $E[T] \leq (1+o(1))n \ln n \frac{1}{p_1 + p_2 \frac{2(1-\xi_2)\mu}{(\mu+1)}}$ *if the quality of each offspring is evaluated*,
2. $E[T] \leq (1+o(1))n \ln n \frac{(1-p_0)}{p_1 + p_2 \frac{2(1-\xi_2)\mu}{(\mu+1)}}$ *if the quality of offspring identical to their parents is not evaluated*; *where*,

$$\xi_i := \frac{p_{i-1,i-2}}{p_{i-1,m} + p_{i-1,i-2} + p_{i-1,i}(1-\xi_{i+1})},$$
$$\xi_m := \frac{p_{m-1,m-2}}{p_{m-1,m} + p_{m-1,m-2}}.$$

The recombination operator of the GA is effective only if individuals with different genotypes are picked as parents (i.e., recombination cannot produce any improvements if two identical individuals are recombined). However, more often than not, the population of the $(\mu+1)$ GA consists only of copies of a single individual. When diversity is created via mutation (i.e., a new genotype is added to the population), it either quickly leads to an improvement or it quickly disappears. The bound on the runtime reflects this behaviour as it is simply a waiting time until one of two event happens; either the current individual is mutated to a better one or diversity emerges and leads to an improvement before it is lost.

The ξ_2 term in the runtime is the conditional probability that once diversity is created by mutation, it will be lost before reaching the next fitness level (an improvement). Naturally, $(1-\xi_2)$ is the probability that a successful crossover will occur before losing diversity. The $(1-\xi_2)$ factor increases with the population size μ and it is independent from the mutation operator used by the algorithms, which implies that larger populations have a higher capacity to maintain diversity long enough to be exploited by the recombination operator. In Fig. 4, how ξ_2 changes with respect to the population size μ is depicted.

Note that setting $p_i := 0$ for all $i > 2$ minimises the upper bound on the expected runtime in the second statement of Theorem 1 and reduces the bound to: $E[T] \leq (1+o(1))n \ln n(p_1+p_2)/\left(p_1 + p_2 \frac{2\mu(1-\xi_2)}{\mu+1}\right)$. Now, we can see the critical role that $\xi^*(\mu) = (1-\xi_2)\mu/(\mu+1)$ plays in the expected runtime. For any population size which yields $\xi^*(\mu) \leq 1/2$, flipping only one bit per mutation becomes advantageous. The best upper bound achievable from the above expression is then $(1+o(1))n \ln n$ by assigning an arbitrarily small constant to p_0, $p_1 = 1-p_0$ (which implies that $p_2 = 0$). As long as $p_0 = \Omega(1)$, when an improvement occurs, the superior genotype takes over the population quickly relative to the time between improvements. Since there are only one-bit flips, the crossover operator becomes virtually useless (i.e., crossover requires a Hamming distance of 2 between parents to create an improving offspring) and the resulting algorithm is a stochastic local search algorithm with a

population. However, when $\xi^*(\mu) > 1/2$, which is the case for all $\mu \geq 5$, setting p_2 as large as possible provides the best upper bound. For $\mu \geq 5$, by setting $p_1 := \epsilon/2$ and $p_0 =: \epsilon/2$ to an arbitrarily small constant ϵ and setting $p_2 = 1 - \epsilon$, we get the upper bound $E[T] \leq (1 + o(1))(1 + \epsilon)n \ln n(\mu + 1)/\big(2\mu\big(1 - \xi_2\big)\big)$, which is plotted for different population sizes in Fig. 1. We have assumed $p_2 = 1$ to obtain the leading constants since the leading constants improve as p_2 approaches 1 and we can pick p_2 to be arbitrarily close to 1.

A direct corollary to the main result is the upper bound for the classical $(\mu + 1)$ GA commonly used in evolutionary computation which applies standard bit mutation with mutation rate c/n for which $p_0 = (1 - o(1))/e^c$, $p_1 = (1 - o(1))c/e^c$ and $p_2 = (1 - o(1))c^2/(2e^c)$.

Corollary 1 *Let ξ_2 be as defined in Theorem 1. The expected runtime $E[T]$ for the $(\mu + 1)$ GA with $\mu = o(\sqrt{\log n})$ using standard bit mutation with mutation rate c/n, $c = \Theta(1)$ to optimise the* OneMax *function is*:

1. $E[T] \leq (1 + o(1))n \ln n \frac{e^c}{c + \frac{c^2\mu}{(\mu+1)}(1-\xi_2)}$ *if the quality of each offspring is evaluated,*
2. $E[T] \leq (1 + o(1))n \ln n \frac{(1-e^{-c})e^c}{c + \frac{c^2\mu}{(\mu+1)}(1-\xi_2)}$ *if the quality of offspring identical to their parents is not evaluated.*

By calculating $\xi^*(\mu) := (1 - \xi_2)\mu/(\mu + 1)$ for fixed values of μ we can determine values of c (i.e., mutation rate) which minimise the leading constant of the runtime bound in Corollary 1. In Fig. 3 we plot the leading constants in the first statement, minimised by picking the appropriate c values (i.e., the ones that minimise the upper bounds) for μ ranging from 5 to 50. All the presented values improve upon the upper bound on the runtime of $1.96n \ln n$ given in [2] for any $\mu \geq 3$ and $\mu = o(\log n/\log\log n)$. All the upper bounds are smaller than $1.7n \ln n$ and clearly decrease with the population size, signifying an at least 60% increase in speed compared to the $en \ln n(1 - o(1))$ lower bound for the same algorithm without the recombination operator [30, 35].

Considering the leading constants in the second statement of Corollary 1, for all population sizes larger than 5, the upper bound for the optimal mutation rate is smaller than the theoretical lower bound on the runtime of unary unbiased black-box algorithms. For population sizes of 3 and 4, $\xi^* = 1/3$ and the expression to be minimised is $(1 - e^{-c})e^c/(c + c^2/3)$. For $c > 0$, this expression has no minimum and is always larger than one. Thus, at least with our technique, a population of size 5 or larger is necessary to prove that the $(\mu + 1)$ GA outperforms stochastic local search and any other unary unbiased optimisation heuristic. The mutation rates which minimise the upper bounds on the expected runtime are provided in Fig. 6 for the variant which evaluates the duplicate offspring and in Fig. 5 for the variant which does not. It can be appreciated that a mutation rate of approximately $1.44/n$ provides better upper bounds than the standard recommended rate of $1/n$, which instead is known to be optimal for the (1+1) EA on OneMax [30, 35]. For comparison the leading constants of the upper bounds for

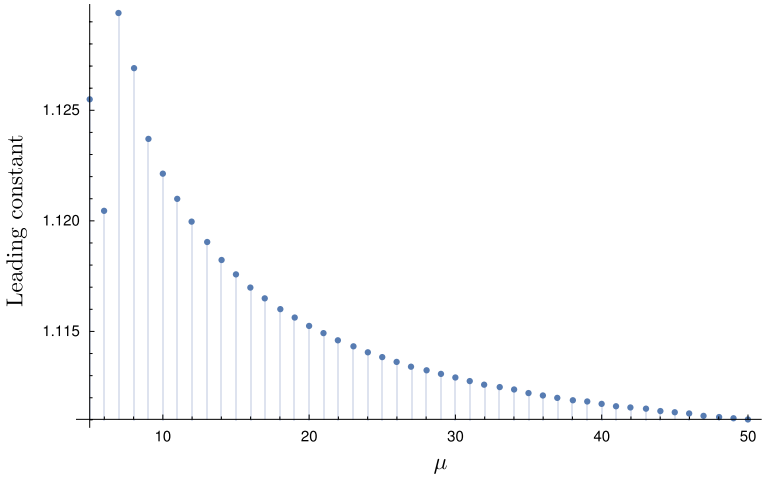

Fig. 7 The leading constant of the upper bound on the runtime when the standard mutation rate $1/n$ is used and the offspring duplicates are not evaluated versus the population size

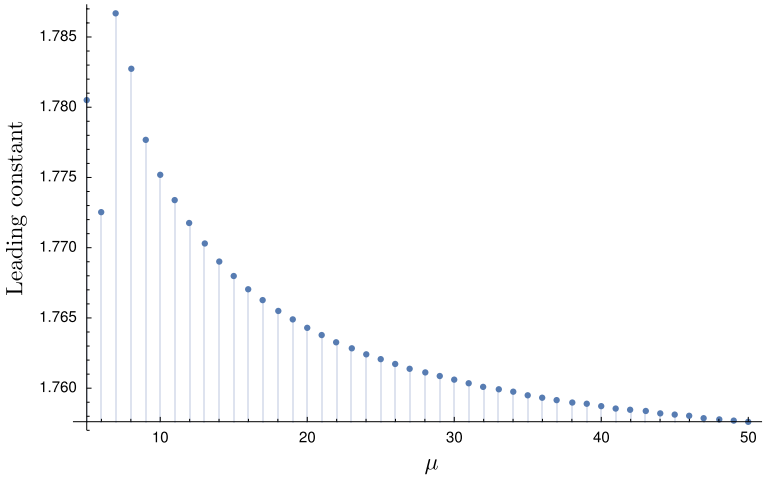

Fig. 8 The leading constant of the upper bound on the runtime when the standard mutation rate $1/n$ is used and the offspring duplicates are evaluated versus the population size

the $(\mu + 1)$ GA with the standard mutation rate of $1/n$ are depicted in Fig. 7 for the variant which does not evaluate duplicate offspring and in Fig. 8 for the variant which does.

4 Analysis

Our main aim is to provide an upper bound on the expected runtime ($E[T]$) of the $(\mu + 1)$ GA defined in Algorithm 1 to maximise the OneMax function. We will provide upper bounds on the expected value $E[T^j]$, where T^j is the time until an individual with at least $j + 1$ one-bits is sampled for the first time given that the initial population consists of individuals with j one-bits (i.e., the population is at level j). Then, by summing up the values of $E[T^j]$ and the expected times for the whole population to reach $j + 1$ one-bits for $j \in \{1 \dots , , n - 1\}$ we achieve a valid upper bound on the expected runtime of the $(\mu + 1)$ GA. Similarly to the analysis in [2], we will pessimistically assume that the algorithm is initialised with all individuals having just zero-bits, and that throughout the optimisation process at most one extra one-bit is discovered at a time.

Although we divide the optimisation process into phases T^j according to the fitness of the population, the widely used artificial fitness levels (AFL) method is not adequate to observe the potential advantages of the recombination operator. The AFL method relies on the minimum probability of improvement given a fitness level and uses its reciprocal to bound the expected time to leave the level from above. Using this minimal probability can give tight results when the probability of improvement is approximately the same for all possible population configurations at a given level. However, when uniform crossover is implemented, the improvement probability does not only depend on the current fitness but it is also heavily dependent on the diversity of the population. Moreover, the above-mentioned diversity is not a scalar value but consists of all pairwise Hamming distances among the individuals in the population, i.e., a $\binom{\mu}{2}$ dimensional variable. In order to avoid working with a multi-dimensional state space we will define the *diversity* $D_t \in \{0, \dots , \lceil \frac{\mu}{2} \rceil - 1\}$, as the number of non-majority individuals in the population at time t, where the *majority genotype* is the genotype that has the most copies in the population. While the improvement probability increases with D_t and can be lower bounded in a way that captures the contribution of crossover, it is no longer straightforward to convert these improvement probabilities to expected runtime values as it is done in the artificial fitness level method. The levels of diversity are different compared to the levels of fitness since while an elitist algorithm never returns to a fitness level it leaves, diversity levels can be visited multiple times and the total time to traverse all the diversity levels cannot be seperated into independent phases.

Fortunately, since the $(\mu + 1)$ GA adds a single solution to the population in every generation, the diversity can change by at most one unless the algorithm finds an improvement. Markov chains of similar structure where the state transition is limited to either moving one state forward or backward, staying put, or moving to the absorbing state can be analysed with existing tools in the literature. However, when we use the possible values for D_t to represent the state of the algorithm we cannot build an explicit Markov chain because, as we discussed above, the transition probabilities would not only depend on D_t but depend on the precise Hamming distances among individuals. This prevents us from having a single transition probability between any two states.

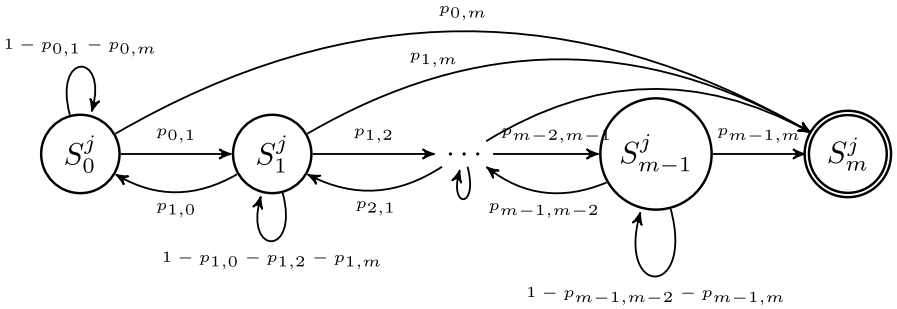

Fig. 9 The topology of Markov Chain M^j

In order to overcome the above-mentioned difficulties in analysing the $(\mu + 1)$ GA on OneMax, we will devise a Markov chain M^j for each $j \in \{0, \dots, n-1\}$ with states that correspond to different levels of diversity in the population. Then, we will analyse the expected absorbing time $E[T_i^j]$ initialised at its state S_i^j by applying Markov chain specific methods (See Definition 1 and Fig. 9). Afterwards, we will prove that M^j is in expectation slower at reaching its absorbing state than the $(\mu + 1)$ GA is at finding an improvement given an initial population at level j. In particular, we will define a non-negative potential function on the domain of all possible configurations of a population at level j or above. Our potential function will be monotonically decreasing with the diversity D_t. Moreover, we will assign to the potential function a value of zero for all populations with at least one solution which has more than j one-bits. Then, we will bound the expected change in the potential function at every iteration (i.e., *the drift*) from below. Using the maximum value of the potential function and the minimum drift, we will derive a bound on the expected time until an improvement is found starting from a population at level j with no diversity (i.e., all the solutions in the population are identical). While this upper bound will not provide an explicit runtime as a function of the problem size, it will allow us to conclude that $(1 + o(1))E[T_0^j] \geq E[T^j]$. Thus, all that remains will be to bound the expected absorbing time of M^j initialised at state S_0^j. We will obtain this bound by identifying the fundamental matrix of M^j. After establishing that the inverse of the fundamental matrix is a *strongly diagonally dominant tridiagonal matrix*, we will make use of existing tools in the literature for inverting such matrices and complete our proof.

4.1 Markov Chain Definition

In this subsection we present the Markov chains which we will use to analyse the behaviour of the $(\mu + 1)$ GA. We should emphasise again that these Markov chains do not represent the exact behaviour of the algorithm and we will later prove that their expected absorbing times can be used to bound the expected time for the actual algorithm to improve the best fitness from $j \in [n-1]$ to $j+1$. Each Markov chain M^j has $m := \lceil \mu/2 \rceil$ transient states $(S_0^j, S_1^j, \dots, S_{m-1}^j)$ and one absorbing state (S_m^j) with the topology depicted in Fig. 9. The absorbing state S_m^j represents a population

with at least one improved individual with more than j one-bits. The states S_i^j represent the amount of diversity in the population. In particular, for $i \in \{0, 1, \ldots, m-2\}$, the state S_i^j denotes populations where all the individuals have j one-bits and all but i of them share the same genotype. More diverse populations at fitness level j, which have at most $\mu - (m-1)$ identical individuals, are all denoted by the last transient state S_{m-1}^j. We have picked $m := \lceil \mu/2 \rceil$ because increasing the number of minority individuals above half of the population cannot be accomplished by copying the existing individuals and these higher diversity levels have significantly smaller probabilities to be observed. Compared to the analysis presented in [2] that used Markov chains of only three states (i.e., no diversity, diversity, increase in one-bits), M^j allows to control the diversity in the population more precisely, thus to show that larger populations are beneficial to the optimisation process.

Definition 1 Let M^j be a Markov chain with $m := \lceil \mu/2 \rceil$ transient states $(S_0^j, S_1^j, \ldots, S_{m-1}^j)$ and one absorbing state (S_m^j) with transition probabilities $p_{i,k}$ from state S_i^j to state S_k^j as follows:

$$
\begin{aligned}
p_{0,1} &:= \frac{\mu}{(\mu+1)} \frac{2j(n-j)p_2}{n^2}, \quad p_{0,m} := \frac{(n-j)p_1}{n}, \\
p_{i,m} &:= 2\frac{i}{\mu}\frac{\mu-i}{\mu}\frac{p_0}{4} \quad \text{if } i > 0, \\
p_{1,2} &:= p_0\left(\left(\frac{1}{\mu}\right)^2 \frac{\mu-1}{\mu+1} + 2\frac{1}{\mu}\frac{\mu-1}{\mu}\frac{1}{4}\frac{\mu-1}{\mu+1}\right), \\
p_{1,0} &:= p_0\left(\left(\frac{\mu-1}{\mu}\right)^2 + 2\frac{1}{\mu}\frac{\mu-1}{\mu}\frac{1}{4}\right)\frac{1}{\mu+1}, \\
p_{i,i+1} &:= p_0\left(\left(\frac{i}{\mu}\right)^2 \min\left(\frac{\mu-i}{\mu+1}, 1/4\right) + 2\frac{i}{\mu}\frac{\mu-i}{\mu}\frac{1}{4}\frac{\mu-i}{\mu+1}\right) \\
&\quad \text{if } m-1 > i > 1, \\
p_{i,i-1} &:= p_0\left(\left(\frac{\mu-i}{\mu}\right)^2 + 2\frac{i}{\mu}\frac{\mu-i}{\mu}\frac{1}{4} + \left(\frac{i}{\mu}\right)^2 \frac{1}{16}\right)\frac{i}{\mu+1} \\
&\quad \text{if } m > i > 1, \\
p_{m,m} &:= 1, \quad p_{0,0} := 1 - p_{0,1} - p_{0,m}, \\
p_{m-1,m-1} &:= 1 - p_{m-1,m-2} - p_{m-1,m}, \\
p_{i,i} &:= 1 - p_{i,i-1} - p_{i,i+1} - p_{i,m} \quad \text{if } 0 < i < m-1, \\
p_{i,j} &:= 0 \quad \textit{otherwise}.
\end{aligned}
$$

Now, we will point out the important characteristics of these transition probabilities. The transition probabilities, $p_{i,k}$, are set to be equal to provable bounds on the probabilities of the $(\mu+1)$ GA with a population consisting of solutions with j bits of gaining/losing diversity ($p_{i,i+1}/p_{i,i-1}$) and sampling a solution with more than j one-bits ($p_{i,m}$). In particular, upper bounds are used for the transition probabilities

$p_{i,k}$ where $i < k$ and lower bounds are used for the transition probabilities $p_{i,k}$ where $i > k$ which will be shown to be precise up to a $(1 + o(1))$ factor in Lemma 3. Note that greater diversity corresponds to a higher probability of two distinct individuals with j one-bits being selected as parents and improved via recombination (i.e., $p_{i,m}$ monotonically increases with i and recombination is ineffective if $i = 0$ and the improvement probability $p_{0,m}$ is simply the probability of increasing the number of one-bits by mutation only. Thus, $p_{0,m} = \Theta((n-j)/n)$ while $p_{i,m} = \Theta(i(\mu - i)/\mu^2)$ when $i > 0$. The first forward transition probability $p_{0,1}$ denotes the probability of the mutation operator of creating a different individual with j one-bits and the selection operator removing one of the majority individuals from the population. The other transition probabilities, $p_{i,i+1}$ and $p_{i,i-1}$ bound the probability that a copy of the minority solution or the majority solution is added to the population and that a member of the other species (minority/majority) is removed in the subsequent selection phase. All transition probabilities except $p_{0,1}$ and $p_{0,m}$ are independent of j and referred to in the theorem statements without specifying j.

4.2 Validity of the Markov Chain Model

In this subsection we will present how we establish that M^j is a pessimistic representation of Algorithm 1 initialised with a population of μ identical individuals at level j. In particular, we will first show that $E[T_0^j]$, the expected absorbing time starting from state S_0^j, is larger than $E[T^j]$. Consequently, this result will allow us to bound the leading constant of the expected runtime of $(\mu + 1)$ GA from above by the leading constant of $\sum_{j=0}^{n-1} E[T_0^j]$ in the following lemma:

Lemma 1 *Let $E[T]$ be the expected runtime until the $(\mu + 1)$ GA with $\mu = o(\log n / \log\log n)$ and $\{p_0, p_1, p_2\} = \Omega(1)$ optimises the* OneMax *function and let*

$E[T_i^j]$ (or $E[T_i]$ wherever j is prespecified) be the expected absorbing time of M^j starting from state S_i^j. Then, $E[T] \leq o(n \log n) + (1 + o(1)) \sum_{j=0}^{n-1} E[T_0^j]$.

The sum $\sum_{j=0}^{n-1} E[T_0^j]$ excludes the fitness evaluations between when the first solution at level j is created and when the whole population is at level j or better. The expectation of the number of omitted evaluations is $O(\mu \log \mu)$ for each fitness level using standard take-over arguments [2]. The restriction of the population size to $\mu = o(\log n / \log\log n)$ allows to bound the expectation of the total number of omitted evaluations by the $o(n \log n)$ term which does not affect the leading constant

We use drift analysis [18], a frequently used tool in the runtime analysis of evolutionary algorithms, to prove the above result. In particular, we will make use of the additive drift theorem from [12], using its formulation from [18].

Theorem 2 (Additive Drift Theorem [12, 18]) *Let $(X_t)_{t \geq 0}$ be a sequence of non-negative random variables with a finite state space $\mathcal{S} \subseteq \mathbb{R}_0^+$ such that $0 \in \mathcal{S}$. Let $T := \inf\{t \geq 0 \mid X_t = 0\}$. If there exists $\delta > 0$ such that for all $s \in \mathcal{S} \setminus \{0\}$* and for all $t \geq 0$,

$$E[\Delta_t(s)] := E[X_t - X_{t+1} \mid X_t = s] \geq \delta,$$

then

$$E[T] \leq \frac{E[X_0]}{\delta}.$$

We will start by defining a potential function over the state space of Alg. 1 that maps a state of the population to the conditional expected absorbing time of M^j that is initialised at the corresponding state in the Markov chain. Note that the drift analysis will be applied to the stochastic process that represents the actual algorithm, not on the Markov chain itself. The Markov chain will only be used to define the potential function that will be used for the drift analysis. The minimum of the potential function will correspond to the state of Algorithm 1 which has sampled a solution with more than j one-bits and we will explicitly prove that the maximum of the potential function is $E[T_0^j]$. Then, we will show that the drift, i.e., the expected decrease in the potential function value in a single time unit (from time t to $t+1$), is at least $1 - o(1)$. Using the maximum value of the potential function and the minimum drift, we will bound the expected time until the algorithm leaves fitness level j, by the absorbing time of the Markov chain M^j.

We will define our potential function over the domain of all possible population diversities at level j. We will refer to the genotype with the most copies in the population as the *majority genotype* and recall that the *diversity*, $D_t \in \{0, \ldots, \mu - 1\}$, of a population P_t is defined as the number of non-majority individuals in the population.

Definition 2 The potential function value for level j, g^j (or g wherever j is prespecified), is defined as follows:

$$g^j(D_t) := g_t^j := \begin{cases} E[T^j_{D_t}] & 0 \leq D_t < \lceil \mu/2 \rceil - 1 \\ E[T^j_{\lceil \mu/2 \rceil - 1}] & \lceil \mu/2 \rceil - 1 \leq D_t < \mu - 1 \\ 0 & \exists x \in P_t \quad s.t. \quad OneMax(x) > j, \end{cases}$$

where $E[T_i^j]$ (denoted as $E[T_i]$ wherever j is prespecified) is the expected absorbing time of the Markov chain M^j starting from state S_i^j.

The absorbing state of the Markov chain corresponds to a population with at least one individual with more than j one-bits, thus having potential function value (and expected absorbing time) equal to zero. The state S_0^j corresponds to a population with no diversity. This potential function is quite similar to the so-called *canonical potential function* [18], which maps each state S_i to the conditional expected runtime given that the process is initialised at S_i. The drift of the canonical potential function is always equal to 1 due to the law of total expectation. The potential function g^j similarly maps states to conditional expectations, however the referred conditional expectations belong to a different but related stochastic process, the Markov chain M^j. Moreover, when $D_t > 0$, the exact transition probabilities for the algorithm cannot be expressed as a function that only

depends on D_t. When diversity is present in the population the transition probabilities depend also on the pairwise Hamming distance between all pairs of individuals in the population. However, bounds on the transition probabilities that hold for any configuration of population for a given D_t can be obtained and these bounds are used to design the Markov chain M^j, which in turn provides us with the conditional expectations that define our potential function. The following lemma formalises that the expected absorbing time gets larger as the initial states get further away from $\lceil \mu/2 \rceil - 1$. The main observation behind this result is that the probability of directly jumping to the absorbing state increases as the process approaches the end of the chain. This property implies that the expected absorbing time from state S_0^j constitutes an upper bound for the potential function g^j.

Lemma 2 *Let $E[T_i^j]$ be the expected absorbing time of the Markov chain M^j conditional on its initial state being S_i^j. Then, $\forall j \in \{0, \ldots, n-1\}$, $E[T_i^j] \leq E[T_{i-1}^j]$ for all $1 \leq i \leq \lceil \mu/2 \rceil$ and $g_t^j \leq E[T_0^j] \, \forall t > 0$.*

Proof We are interested in $\max(E[T_0], E[T_1], \ldots, E[T_m])$ since these are the values that the potential function g can have. According to the transition probabilities in Definition 1, $p_{i+1,m} \geq p_{i,m}$ for all i. Using this observation and the law of total expectation we will show that not only $\max(E[T_0], E[T_1], \ldots, E[T_m]) = E[T_0]$ but also $E[T_{i-1}] \geq E[T_i]$ for all i. First, we will prove that $E[T_{m-2}] \geq E[T_{m-1}]$ by contradiction. Then, we will prove by induction that $E[T_{i-1}] \geq E[T_i]$ for all i. For this induction we will use $E[T_{m-2}] \geq E[T_{m-1}]$ as our basic step and we will prove by contradiction that if for all $k > i$, $E[T_{k-1}] \geq E[T_k]$ holds, then $E[T_{i-1}] \geq E[T_i]$ must also hold.

If we use the law of total expectation for the absorbing time starting from state $0 < i < m$, we obtain:

$$\begin{aligned} E[T_i] =& p_{i,i+1}(E[T_{i+1}] + 1) + p_{i,i-1}(E[T_{i-1}] + 1) \\ &+ p_{i,m} + (1 - p_{i,i+1} - p_{i,i-1} - p_{i,m})(E[T_i] + 1). \end{aligned}$$

This equation can be rearranged as follows:

$$1 = p_{i,i+1}(E[T_i] - E[T_{i+1}]) + p_{i,i-1}(E[T_i] - E[T_{i-1}]) + p_{i,m}E[T_i].$$

For the special case of $i = m - 1$, we have:

$$1 = p_{m-1,m}(E[T_{m-1}]) + p_{m-1,m-2}(E[T_{m-1}] - E[T_{m-2}]).$$

If we introduce the allegedly contradictory assumption $E[T_{m-2}] < E[T_{m-1}]$, the above equation implies:

$$\begin{aligned} 1 > p_{m-1,m}(E[T_{m-1}]) \implies \frac{1}{p_{m-2,m}} \geq \frac{1}{p_{m-1,m}} \\ > E[T_{m-1}] > E[T_{m-2}] \implies \frac{1}{p_{m-2,m}} > E[T_{m-2}]. \end{aligned}$$

Given that $\frac{1}{p_{i,m}} > E[T_i]$ and $E[T_{i+1}] > E[T_i]$ the law of total expectation for i implies:

$$\begin{aligned}
1 &= p_{i,i+1}(E[T_i] - E[T_{i+1}]) + p_{i,i-1}(E[T_i] - E[T_{i-1}]) + p_{i,m}E[T_i] \\
1 &< p_{i,i+1}(E[T_i] - E[T_{i+1}]) + p_{i,i-1}(E[T_i] - E[T_{i-1}]) + 1 \\
0 &< p_{i,i-1}(E[T_i] - E[T_{i-1}]) \\
E[T_{i-1}] < E[T_i] &\implies \frac{1}{p_{i-1,m}} \geq \frac{1}{p_{i,m}} > E[T_i] > E[T_{i-1}].
\end{aligned}$$

Thus, the allegedly contradictory claim $E[T_{m-1}] > E[T_{m-2}]$ induces over i such that it implies $E[T_1] > E[T_0]$ and $1/p_{0,m} > E[T_0]$. We can now write the total law of expectation for $i = 0$.

$$\begin{aligned}
1 &= p_{0,1}(E[T_0] - E[T_1]) + p_{0,m}E[T_0] \\
1 &< p_{0,1}(E[T_0] - E[T_1]) + 1 \\
0 &< p_{0,1}(E[T_0] - E[T_1]) \\
0 &> p_{0,1}
\end{aligned}$$

The last statement is a contradiction since a probability cannot be negative. This contradiction proves the initial claim $E[T_{m-2}] \geq E[T_{m-1}]$.

We will now follow a similar route to prove that $E[T_{i-1}] \geq E[T_i]$ for all i. Given that for all $k > i$, $E[T_{k-1}] \geq E[T_k]$ holds, we will show that $E[T_i > E[T_{i-1}]$ creates a contradiction. We start with the law of total expectation for $E[T_i]$:

$$1 = p_{i,i+1}(E[T_i] - E[T_{i+1}]) + p_{i,i-1}(E[T_i] - E[T_{i-1}]) + p_{i,m}E[T_i]$$

Our assumption "$\forall k > i : \quad E[T_k] \leq E[T_{k-1}]$" implies that $E[T_i] - E[T_{i+1}] \geq 0$, thus we obtain:

$$1 > p_{i,i-1}(E[T_i] - E[T_{i-1}]) + p_{i,m}E[T_i].$$

With our allegedly contradictory assumption $E[T_i] - E[T_{i-1}] > 0$ we obtain:

$$1 > p_{i,m}E[T_i] \implies \frac{1}{p_{i-1,m}} \geq \frac{1}{p_{i,m}} > E[T_i] > E[T_{i-1}].$$

We have already shown above that $1/p_{i-1,m} \geq 1/p_{i,m} > E[T_i] > E[T_{i-1}]$ can be induced over i and implies $E[T_1] > E[T_0]$ and $1/p_{0,m} > E[T_0]$. Then we can conclude that:

$$E[T_i] \geq E[T_{i+1}] \quad \forall 0 \leq i \leq \lceil \mu/2 \rceil - 1. \tag{1}$$

The above conclusion implies that $E[T_0]$ is the largest value that our potential function can have and $E[T_i] - E[T_{i+1}]$ is non-negative for all i. □

Now that the potential function is bounded from above, we will bound the drift $E[g_t - g_{t+1} \mid D_t = i]$. Due to the law of total expectation, the expected absorbing time, $E[T_i]$ satisfies $\sum_j p_{i,j}(E[T_j] - E[T_i]) = 1$ for any absorbing Markov chain at a transient state i. Since $E[T_i]$ and $E[T_j]$ are the respective potentials of the states S_i

and S_j, the left hand side of the equation closely resembles the drift. In particular, the actual drift at state S_i is also a linear combination of the terms $(E[T_i] - E[T_j])$, although the precise weights (i.e., transition probabilities) of these terms cannot be expressed as functions of i and j alone. However, these weights can be bounded and using these bounds the drift can be compared with the expression $\sum_j p_{i,j}(E[T_j] - E[T_i])$ in order to determine how close it is to 1. Since the probabilities for M^j are pessimistically set to underestimate the drift, we can formally prove the following lemma.

Lemma 3 *For a population at level j,* $E[g_t^j - g_{t+1}^j \mid D_t = i] \geq 1 - o(1)$ *for all* $t > 0$ *and* $i \in \{0, 1, \ldots, \mu - 1\}$.

Proof When there is no diversity in the population (i.e., $D_t = 0$ and $g_t = g(0) = E[T_0]$) the only way to increase the diversity is to introduce it during a mutation operation. A non-majority individual is obtained when one of the $n - j$ zero-bits and one of the j one-bits are flipped while no other bits are touched. Then, one of the majority individuals must be removed from the population during the selection phase. This event has probability at least $p_2 \cdot 2 \cdot \frac{(n-j)}{n} \frac{j}{n} \frac{\mu}{\mu+1} = p_{0,1}$ and decreases the potential to $g(1) = E[T_1]$. Another way to change the potential function value is to create an improved individual with the mutation operator. In order to improve a solution it is sufficient to pick one of $n - j$ one-bits and flip no other bits. This event has probability at least $p_1 \cdot \frac{(n-j)}{n} = p_{0,m}$ and reduces the potential to 0 since an improvement has been found. Thus, we can conclude that when $D_t = 0$, the conditional drift $E[\Delta_t \mid D_t = 0]$ is at least $p_{0,m}(g(0) - 0) + p_{0,1}(g(0) - g(1)) = p_{0,m}E[T_0] + p_{0,1}(E[T_0] - E[T_1])$. The law of total expectation for the state S_0^j of Markov chain M^j similarly states $\sum_j p_{i,j}(E[T_0] - E[T_j]) = p_{0,m}(E[T_0] - E[T_m]) + p_{0,1}(E[T_0] - E[T_1]) = 1$. Since the state S_m^j is the absorbing state, $E[T_m] = 0$, and therefore,

$$E[\Delta_t \mid D_t = 0] \geq p_{0,m}E[T_0] + p_{0,1}(E[T_0] - E[T_1]) = 1.$$

For $D_t > 0$, we will condition the drift on whether the picked parents are both majority individuals $\mathcal{E}_1$, are both minority individuals with the same genotype $\mathcal{E}_2$, are a pair that consists of one majority and one minority individual $\mathcal{E}_3$, or they are both minority individuals with different genotypes $\mathcal{E}_4$.

Let $\mathcal{E}^*$ be the event that the population P_t consists of two genotypes with Hamming distance two. Then,

$$\begin{aligned}
\mathbb{P}\{\mathcal{E}_1 \mid \mathcal{E}^*\} &= \mathbb{P}\{\mathcal{E}_1 \mid \overline{\mathcal{E}^*}\} = \left(\frac{\mu - i}{\mu}\right)^2 \\
\mathbb{P}\{\mathcal{E}_2 \mid \mathcal{E}^*\} &= \mathbb{P}\{\mathcal{E}_2 \mid \overline{\mathcal{E}^*}\} + \mathbb{P}\{\mathcal{E}_4 \mid \overline{\mathcal{E}^*}\} = \left(\frac{i}{\mu}\right)^2 \\
\mathbb{P}\{\mathcal{E}_3 \mid \mathcal{E}^*\} &= \mathbb{P}\{\mathcal{E}_3 \mid \overline{\mathcal{E}^*}\} = 2\frac{i}{\mu}\frac{\mu - i}{\mu}; \quad \mathbb{P}\{\mathcal{E}_4 \mid \mathcal{E}^*\} = 0.
\end{aligned} \tag{2}$$

Let $E[\Delta_t^{i>0}]$ be the drift conditional on $i > 0$. The law of total expectation states:

$$\begin{aligned}
E[\Delta_t^{i>0}] =& \mathbb{P}\{\mathcal{E}^*\} \sum_{k=1}^{4} \mathbb{P}\{\mathcal{E}_k \mid \mathcal{E}^*\} E[\Delta_t^{i>0} \mid \mathcal{E}_k, \mathcal{E}^*] \\
&+ (1 - \mathbb{P}\{\mathcal{E}^*\}) \sum_{k=1}^{4} \mathbb{P}\{\mathcal{E}_k \mid \overline{\mathcal{E}^*}\} E[\Delta_t^{i>0} \mid \mathcal{E}_k, \overline{\mathcal{E}^*}] \\
\geq& \left(\frac{\mu - i}{\mu}\right)^2 E[\Delta_t^{i>0} \mid \mathcal{E}_1] + \left(\frac{i}{\mu}\right)^2 \min\left(E[\Delta_t^{i>0} \mid \mathcal{E}_2], E[\Delta_t^{i>0} \mid \mathcal{E}_4]\right) \\
&+ 2\frac{\mu - i}{\mu}\frac{i}{\mu} E[\Delta_t^{i>0} \mid \mathcal{E}_3],
\end{aligned}$$

where the last inequality is obtained by substituting the probabilities from (2) and rearranging the terms. Due to the Definition 2, the conditional drifts in the above expression depend on the values of $(E[T_i] - E[T_j])$ for $(i,j) \in \{0, \ldots, m\}$. Therefore, in order to prove a lower bound on the drift, we will make use of another equality where the $(E[T_i] - E[T_j])$ terms appear. We will now write the law of total expectation for state i for our Markov chain M^j and then replace the probabilities in the law of total expectation with the values from Definition 1 and rearrange it into additive terms with the probabilities of events $\mathcal{E}_i$ as multiplicative factors.

$$\begin{aligned}
1 &= p_{i,i+1}(E[T_i] - E[T_{i+1}]) + p_{i,i-1}(E[T_i] - E[T_{i-1}]) + p_{i,m}E[T_i] \\
&= \left(\frac{\mu - i}{\mu}\right)^2 p_0 \frac{i}{\mu + 1}(E[T_i] - E[T_{i-1}]) \\
&\quad + \left(\frac{i}{\mu}\right)^2 p_0 \left(\min\left(\frac{\mu - i}{\mu + 1}, 1/4\right)(E[T_i] - E[T_{i+1}]) + \frac{i(E[T_i] - E[T_{i-1}])}{16(\mu + 1)}\right) \\
&\quad + 2\frac{i}{\mu}\frac{\mu - i}{\mu} p_0 \left(\frac{1}{4}\frac{\mu - i}{\mu + 1}(E[T_i] - E[T_{i+1}]) + \frac{i(E[T_i] - E[T_{i-1}])}{4(\mu + 1)} + \frac{E[T_i]}{4}\right).
\end{aligned} \tag{3}$$

In particular, we will prove that the following three inequalities hold in order to prove that $E[\Delta_t^{i>0}] \geq 1 - o(1)$.

$$E[\Delta_t^{i>0} \mid \mathcal{E}_1] \geq (1 - o(1)) \cdot p_0 \frac{i}{\mu + 1}(E[T_i] - E[T_{i-1}]) \tag{4}$$

$$\begin{aligned}
&\min\left(E[\Delta_t^{i>0} \mid \mathcal{E}_2], E[\Delta_t^{i>0} \mid \mathcal{E}_4]\right) \geq (1 - o(1)) \cdot \\
&p_0 \left(\min\left(\frac{\mu - i}{\mu + 1}, 1/4\right)(E[T_i] - E[T_{i+1}]) + \frac{1}{16}\frac{i}{\mu + 1}(E[T_i] - E[T_{i-1}])\right)
\end{aligned} \tag{5}$$

$$E[\Delta_t^{i>0} \mid \mathcal{E}_3] \geq (1 - o(1)) \cdot$$
$$p_0 \left(\frac{1}{4} \frac{\mu - i}{\mu + 1} (E[T_i] - E[T_{i+1}]) + \frac{1}{4} \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) + \frac{E[T_i]}{4} \right) \tag{6}$$

We will start with (4).When two majority individuals are selected as parents ($\mathcal{E}_1$), we pessimistically assume that improving to the next level and increasing the diversity has zero probability. Losing the diversity requires that no bits are flipped during mutation (with probability p_0) and that a minority individual will be removed from the population with probability $\frac{i}{\mu+1}$. Therefore, $E[\Delta_t^{i>0} \mid \mathcal{E}_1] \geq p_0 \big(E[T_i] - E[T_{i-1}]\big) \frac{i}{\mu+1}$ which proves that (4) holds.

Next, we will prove the inequality (5). When two minority individuals are selected as parents ($\mathcal{E}_2$ or $\mathcal{E}_4$), if they are identical ($\mathcal{E}_2$) then it is sufficient that the mutation does not flip any bits and that a majority individual is removed from the population. Thus, given $\mathcal{E}_2$, the probability of increasing the diversity is at least $p_0 \cdot (\mu - i)/(\mu + 1)$ and the probability of creating a majority individual is $\mathcal{O}\big(1/n^2\big)$ since it is necessary to flip at least two particular bit positions. The resulting lower bound is,

$$\begin{aligned} E[\Delta_t^{i>0} \mid \mathcal{E}_2] &\geq p_0 \frac{\mu - i}{\mu + 1} (E[T_i] - E[T_{i+1}]) + \mathcal{O}\Big(\frac{1}{n^2}\Big)(E[T_i] - E[T_{i-1}]) \\ &\geq (1 - o(1)) \cdot p_0 \frac{\mu - i}{\mu + 1} (E[T_i] - E[T_{i+1}]). \end{aligned} \tag{7}$$

If the two minority individuals have a Hamming distance of $2d \geq 2$ (i.e., $\mathcal{E}_4$), then in order to create another minority individual at the end of the crossover operation it is sufficient that crossover picks exactly d one-bits and d zero-bits among $2d$ bit positions where they differ. There are $\binom{2d}{d}$ different ways that this can happen and the probability that any particular outcome of crossover is realised is 2^{-2d}. One of those outcomes though, might be the majority individual and if that is the case the diversity can decrease afterwards. However, while the Hamming distance between the minority individuals can be $2d = 2$, obtaining a majority individual by recombining two minority individuals requires at least four specific bit positions to be picked correctly during the crossover with probability at most 1/16 or at least one specific bit must be flipped by the mutation operator to obtain the majority individual with probability at most $\mathcal{O}(1/n)$. On the other hand, when two different minority individuals are selected as parents, there is at least a $\frac{1 - \binom{2d}{d} 2^{-2d}}{2}$ probability that the crossover will result in an individual with more one-bits and then with probability p_0 the mutation will not flip any bits. The probability $\frac{1 - \binom{2d}{d} 2^{-2d}}{2}$ monotonically increases with d and has its minimum value 1/4 for $d = 1$. Hence,

$$
\begin{aligned}
E[\Delta_t^{i>0} \mid \mathcal{E}_4] &\geq p_0 \Bigg[\left(\binom{2d}{d} - 1 \right) 2^{-2d} \frac{\mu - i}{\mu + 1} (E[T_i] - E[T_{i+1}]) \\
&\quad + \left(\min\left(\frac{1}{16}, 2^{-2d} \right) + \mathcal{O}(1/n) \right) \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) + \frac{1}{4} E[T_i] \Bigg] \\
&\geq (1 - o(1)) \cdot p_0 \left[\frac{1}{16} \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) + \frac{E[T_i]}{4} \right]. \\
&\geq (1 - o(1)) \cdot p_0 \left[\frac{1}{16} \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) + \frac{1}{4} (E[T_i] - E[T_{i-1}]) \right].
\end{aligned} \tag{8}
$$

Where in the last inequality we used the fact that $E[T_i] > E[T_i] - E[T_{i+1}]$. The inequalities (7) and (8) establish that (5) holds. Finally, we will prove that (6) holds. We will consider the drift conditional on event $\mathcal{E}_3$, the case when one minority and one majority individual are selected as parents. We will further divide this event into two subcases. In the first case the Hamming distance $2d$ between the minority and the majority individual is exactly two ($d = 1$). Then, the probabilities that crossover creates a copy of the minority individual, a copy of the majority individual or a new individual with more one-bits are all equal to 1/4 unless the mutation operator flips at least one of the bit positions where the minority and majority individuals differ which happens with probability at most $\mathcal{O}(1/n)$ which we can absorb in the $(1 - o(1))$ multiplier. Thus, the conditional drift is:

$$
\begin{aligned}
&E[\Delta_t^{i>0} \mid \mathcal{E}_3, d = 1] \\
&\quad \geq (1 - o(1)) \frac{p_0}{4} \left(\frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) + \frac{\mu - i}{\mu + 1} (E[T_i] - E[T_{i+1}]) + E[T_i] \right).
\end{aligned} \tag{9}
$$

On the other hand, when $d > 1$, the drift is more similar to the case of $\mathcal{E}_4$ where the probabilities of creating copies of either the minority or the majority individuals diminish with larger d while larger d increases the probability of creating an improved individual. More precisely,

$$
\begin{aligned}
E[\Delta_t^{i>0} \mid \mathcal{E}_3, d > 1] &\geq p_0 \Bigg(\left(\binom{2d}{d} - 1 \right) 2^{-2d} \frac{\mu - i}{\mu + 1} (E[T_i] - E[T_{i+1}]) \\
&\quad + \left(2^{-2d} + \mathcal{O}(1/n) \right) \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) + \left(\frac{1 - \binom{2d}{d} 2^{-2d}}{2} \right) E[T_i] \Bigg)
\end{aligned}
$$

We will now show that $E[\Delta_t^{i>0} \mid \mathcal{E}_3, d > 1] \geq E[\Delta_t^{i>0} \mid \mathcal{E}_3, d = 1]$ which together with (9) will imply that (6) holds. Since $(E[T_i] - E[T_{i-1}])$ is negative, for $d > 1$,

$$
\left(2^{-2d} + \mathcal{O}(1/n) \right) \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]) \geq \frac{1}{4} \frac{i}{\mu + 1} (E[T_i] - E[T_{i-1}]).
$$

As the multiplier of the negative $(E[T_i] - E[T_{i-1}])$ term decreases when $d > 1$, showing that the sum of the remaining two positive terms increases when $d > 1$, establishes that $E[\Delta_t^{i>0} \mid \mathcal{E}_3, d > 1] \geq E[\Delta_t^{i>0} \mid \mathcal{E}_3, d = 1]$. Between the cases of $d = 1$ and $d > 1$, the probability of creating a copy of the minority individual decreases from 1/4 to $\left(\binom{2d}{d} - 1\right)2^{-2d}$ while the probability of creating an improvement increases from 1/4 to $\left(\frac{1-\binom{2d}{d}2^{-2d}}{2}\right)$. A trivial but crucial observation here is that $\frac{\mu-i}{\mu+1}(E[T_i] - E[T_{i+1}]) < E[T_i]$. Thus, if,

$$\left(\left(\binom{2d}{d} - 1\right)2^{-2d} - \frac{1}{4}\right) + \left(\left(\frac{1 - \binom{2d}{d}2^{-2d}}{2}\right) - \frac{1}{4}\right) > 0$$

then, $E[\Delta_t^{i>0} \mid \mathcal{E}_3, d > 1] \geq E[\Delta_t^{i>0} \mid \mathcal{E}_3, d = 1]$. We can factorise the above expression as,

$$\left(\left(\binom{2d}{d} - 1\right)2^{-2d} - \frac{1}{4}\right) + \left(\frac{1 - \binom{2d}{d}2^{-2d}}{2}\right) - \frac{1}{4} = 2^{-1-2d}\left(\binom{2d}{d} - 2\right)$$

where we can see that it is positive as long as $\binom{2d}{d} > 2$, which holds for all $d \geq 1$ and proves that $E[\Delta_t^{i>0} \mid \mathcal{E}_3, d > 1] \geq E[\Delta_t^{i>0} \mid \mathcal{E}_3, d = 1]$. This in turn shows that (6) holds and establishes our claim.

□

Now, we can prove Lemma 1 using the above results.

Proof of Lemma 1 Since $E[g_t^j - g_{t+1}^j \mid D_t = i] \geq 1 - o(1)$ from Lemma 3 and $g_t^j \leq E[T_0^j]$ from Lemma 2, we can apply Theorem 2 to obtain $E[T^j] \leq (1 + o(1))E[T_0^j]$. Given that there are k individuals with at least $j + 1$ one-bits (*improved individuals*) in the population, in every generation with probability at least k/μ, at least one improved individual is selected as parent. If we pessimistically assume that the other parent has only j 1-bits, the Hamming distance between the genotypes is equal to $2d + 1$ for some integer $d > 0$. Let ℓ be an arbitrarily picked bit-position which was set to 1 in the improved individuals and to 0 in the other parents. With probability 1/2, this bit-position will be set to 1 in the offspring as well. For the remaining $2d$ bit-positions, the probability that at least d 1-bits are inherited by the offspring is at least 1/2 due to the symmetry of the actual outcome around d. Thus, given that there are k improved individuals in the population, the expected

time until a new improved individual is at most $\mu/k \cdot 8 \cdot (1-1/n)^{-n} = O(\mu/k)$. Summing over $k \in [\mu]$ we obtain, $\sum_{k=1}^{\mu} O(\mu/k) = O(\mu \log \mu)$ expected iterations after sampling the first improved solution, as an upper bound on the expected time until all individuals in the population have at least $j+1$ 1-bits. If the population size is in the order of $o(\log n/\log\log n)$, then the total number of iterations where there are individuals with different fitness values in the population is in the order of $o(n \log n)$. Since $j \in \{0, 1, \ldots, n-1\}$, we can establish that

$$E[T] \leq o(n \log n) + \sum_{j=0}^{n-1} E[T^j] \leq o(n \log n) + (1 + o(1)) \sum_{j=0}^{n-1} E[T_0^j].$$

□

4.3 Markov Chain Absorption Time Analysis

In the previous subsection we stated in Lemma 1 that we can bound the absorbing times of the Markov chains M^j to derive an upper bound on the runtime of Algorithm 1. In this subsection we use mathematical tools developed for the analysis of Markov chains to provide such bounds on the absorbing times.

The absorbing time of a Markov chain starting from any initial state i can be derived by identifying its fundamental matrix. Let the matrix Q denote the transition probabilities between the transient states of the Markov chain M^j. The fundamental matrix of M^j is defined as $N := (I-Q)^{-1}$ where I is the identity matrix. The most important characteristic of the fundamental matrix is that, when it is multiplied by a column vector of ones, the product is a vector holding $E[T_i^j]$, the expected absorbing times conditional on the initial state i of the Markov chain. Since, Lemma 1 only involves T_0^j, we are only interested in the entries of the first row of $N = [n_{ik}]$. However, inverting the matrix $I-Q$ is not always a straightforward task. Fortunately, $I-Q = [a_{ik}]$ has characteristics that allow bounds on the entries of its inverse. Its entries are related to the transition probabilities of M^j as follows:

$$a_{11} = 1 - p_{0,0} = p_{0,1} + p_{0,m} \tag{10}$$

$$a_{mm} = 1 - p_{m-1,m-1} = p_{m-1,m-2} + p_{m-1,m} \tag{11}$$

$$\begin{aligned} a_{ii} &= 1 - p_{i-1,i-1} = p_{i-1,i-2} + p_{i-1,i} + p_{i-1,m} \\ &\forall i \in \{2, ..., m-1\} \end{aligned} \tag{12}$$

$$a_{ik} = -p_{i-1,k-1} \quad \forall i,k \in \{1, \ldots, m\} \wedge i \neq k \tag{13}$$

Observe that $I-Q$ is a *tridiagonal* matrix, in the sense that all non-zero elements of $I-Q$ are either on the diagonal or adjacent to it. Moreover, the diagonal entries a_{ii} of $I-Q$ are in the form $1-p_{i-1,i-1}$, which is equal to the sum of all transition probabilities out of state $i-1$. Since the other entries on row i are

transition probabilities from state $i-1$ to adjacent states, we can see that $|a_{ii}| > \sum_{i\neq k} |a_{ik}|$. The matrices where $|a_{ii}| > \sum_{i\neq k} |a_{ik}|$ holds are called *strongly diagonally dominant* (SDD). Since $I-Q$ is SDD, according to [21, Lemma 2.1], it holds for the fundamental matrix N for all $i \neq k$ that, $|n_{i,k}| \leq |n_{k,k}| \leq \left(|a_{k,k}|\left(1-\frac{\sum_{l\neq k}|a_{l,k}|}{|a_{k,k}|}\right)\right)^{-1} \leq \left(|a_{k,k}| - \sum_{l\neq k}|a_{l,k}|\right)^{-1}$.

In our particular case, the above inequality implies that $|n_{1,k}| \leq 1/p_{k-1,m}$. For any population with diversity, there is a probability in the order of $\Omega(1/\mu)$ of selecting one minority and one majority individual and a constant probability that their offspring will have more one-bits than the current level. Considering $m = \mathcal{O}(\mu)$, we obtain:

$$E[T_0^j] = \sum_{k=1}^{m} n_{1,k} < n_{1,1} + \sum_{k=2}^{m} \frac{1}{p_{k-1,m}} \leq n_{1,1} + \mathcal{O}\left(\mu^2\right). \tag{14}$$

We note here that the $\mathcal{O}\left(\mu^2\right)$ factor in the above expression creates the condition $\mu = o(\sqrt{\log n})$ on the population size for our main results. We will now bound the term $n_{1,1}$ from above to establish our upper bound using the following theorem which follows from [21, Corollary 3.2]:

Theorem 3 *Let A be an $m \times m$ tridiagonal non-singular SDD matrix such that $a_{i,k} \leq 0$ for all $i \neq k$, $A^{-1} = [n_{i,k}]$ exists and $n_{i,k} \geq 0$ for all i, k. Then, $n_{1,1} = 1/(a_{1,1} + a_{1,2}\xi_2)$, where $\xi_i = a_{i,i-1}/(a_{i,i} + a_{i,i+1}\xi_{i+1})$, and $\xi_m = a_{m,m-1}/a_{m,m}$.*

In order to use Theorem 3, we need to satisfy its conditions. We can easily see that non-diagonal entries of the original matrix $I-Q$ are non-positive and use [21, Theorem 3.1] to show that $N = (I-Q)^{-1}$ has no negative entries:

Theorem 4 (Theorem 3.1 in [21]) *If A is a tridiagonal non-singular SDD matrix and $a_{i,i} > 0$, then $A^{-1} = [n_{i,k}]$ exists and*

- $sign(n_{i,i}) = 1$
- $sign(n_{i,k}) = (-1)^{i+k} \prod_{l=k+1}^{i} a_{l,l-1}, i > k$
- $sign(n_{i,k}) = (-1)^{i+k} \prod_{l=i}^{k} a_{l,l+1}, i < k.$

Since the diagonal entries of $I-Q$ are strictly positive, according to Theorem 4 the diagonal entries of N are also positive. The non-diagonal entries of $I-Q$ are all negative thus the series multiplication from Theorem 4 for $i > k$ reduces to $(-1)^{i+k+i-k} = (-1)^{2i} = 1$. Similarly for the case $i < k$, the multiplication reduces to $(-1)^{i+k+k-i} = (-1)^{2k} = 1$. Hence, N does not have any negative entries.

Lemma 4 *With an initial population of size $\mu = o(\sqrt{\log n})$ at level j, the expected time $E[T^j]$ until an individual with $j+1$ one-bits is sampled by the $(\mu+1)$ GA for the first time is bounded from above as follows*:

$$E[T_0^j] \leq \frac{n}{n-j} \frac{1}{p_1 + \frac{2\mu p_2}{(\mu+1)} \frac{j}{n}(1-\xi_2)} + o(\log n), \text{ where,}$$
$$\xi_m = \frac{p_{m-1,m-2}}{p_{m-1,m} + p_{m-1,m-2}}$$
$$\xi_i = \frac{p_{i-1,i-2}}{p_{i-1,m} + p_{i-1,i-2} + p_{i-1,i}(1-\xi_{i+1})} \quad \forall 1 < i < m = \lceil \mu/2 \rceil.$$

where $p_{i,k}$ are the transition probabilities of the Markov chain M^j.

Proof Starting from Inequality 14 and applying Theorem 3 we obtain:

$$\begin{aligned}
E[T_0^j] &\leq n_{1,1} + \mathcal{O}(\mu^2) \leq \frac{1}{a_{1,1} + a_{1,2}\xi_2} + \mathcal{O}(\mu^2) \\
&= \frac{1}{p_{0,m} + p_{0,1}(1-\xi_2)} + \mathcal{O}(\mu^2) \\
&\leq \frac{1}{p_{0,m} + p_{0,1}(1-\xi_2)} + \mathcal{O}(\mu^2) \\
&\leq \frac{1}{\frac{(n-j)p_1}{n} + \frac{\mu}{(\mu+1)} \frac{2j(n-j)p_2}{n^2}(1-\xi_2)} + \mathcal{O}(\mu^2) \\
&= \frac{n}{n-j} \frac{1}{p_1 + \frac{\mu}{(\mu+1)} \frac{2jp_2}{n}(1-\xi_2)} + o(\log n),
\end{aligned}$$

where we substitute $p_{0,1}$ and $p_{0,m}$ with their values declared in Definition 1. The definitions of ξ_i and ξ_m are obtained by simply substituting the matrix entries in Theorem 3 with their respective values from Eqs. 10 to 13. □

In Lemma 4, we can now see that the bound depends on p_1 and p_2 of the mutation operator and ξ_2. The term ξ_i represents the conditional probability of returning to the state $(i-1)$ given that you initialise at state i. It only depends on the probabilities of creating either a copy of the minority/majority solution or an improvement. Since we pessimistically assume that we can only improve by crossover when diversity is present, the leading term of all these probabilities include p_0. Thus, when the conditional probabilities ξ_i are calculated, the common factor p_0 disappears and we obtain the upper bound that only depends on p_1, p_2 and μ.

The above bound on $E[T_0^j]$, together with Lemma 1, yields Theorem 1, our main result.

Proof of Theorem 1 Combining Lemmas 1 and 4 we obtain:

$$
\begin{aligned}
E[T] &\leq o(n \log n) + (1 + o(1)) \sum_{j=0}^{n-1} E[T^j] \\
&\leq o(n \log n) + (1 + o(1)) \sum_{j=0}^{n-1} \left(\frac{n}{n-j} \frac{1}{p_1 + \frac{\mu}{(\mu+1)} \frac{2jp_2}{n} (1 - \xi_2)} \right)
\end{aligned} \tag{15}
$$

We will now divide the sum into two smaller sums:

$$
\begin{aligned}
&\sum_{j=1}^{n} \frac{n}{n-j} \frac{1}{p_1 + \frac{\mu}{(\mu+1)} \frac{2jp_2}{n} (1 - \xi_2)} \\
&= \sum_{j=1}^{n - n/\sqrt{\log n}} \frac{n}{n-j} \frac{1}{p_1 + \frac{\mu}{(\mu+1)} \frac{2jp_2}{n} (1 - \xi_2)} \\
&\quad + \sum_{j=n-n/\sqrt{\log n}+1}^{n} \frac{n}{n-j} \frac{1}{p_1 + \frac{\mu}{(\mu+1)} \frac{2jp_2}{n} (1 - \xi_2)} \\
&\leq \mathcal{O}\left(n\sqrt{\log n} \right) + \frac{n}{p_1 + \frac{2p_2\mu}{(\mu+1)} \left(1 - \frac{1}{\sqrt{\log n}} \right) (1 - \xi_2)} \sum_{j=n-n/\sqrt{\log n}+1}^{n} \frac{1}{n-j} \\
&\leq \mathcal{O}\left(n\sqrt{\log n} \right) + \frac{n \ln n}{p_1 + \frac{2p_2\mu}{(\mu+1)} \left(1 - \frac{1}{\sqrt{\log n}} \right) (1 - \xi_2)}
\end{aligned}
$$

We conclude the proof by substituting the sum in Eq. 15 with the above expression.

$$
\begin{aligned}
E[T] &\leq o(n \log n) \\
&\quad + (1 + o(1)) \left(\mathcal{O}\left(n\sqrt{\log n} \right) + \frac{n \ln n}{p_1 + \frac{2p_2\mu}{(\mu+1)} \left(1 - \frac{1}{\sqrt{\log n}} \right) (1 - \xi_2)} \right) \\
&\leq o(n \log n) \\
&\quad + (1 + o(1)) \left(\mathcal{O}\left(n\sqrt{\log n} \right) + (1 + o(1)) \frac{n \ln n}{p_1 + \frac{2p_2\mu}{(\mu+1)} (1 - \xi_2)} \right) \\
&= (1 + o(1)) n \ln n \frac{1}{p_1 + \frac{2p_2\mu}{(\mu+1)} (1 - \xi_2)}.
\end{aligned}
$$

The expressions for ξ_i and ξ_m come from Lemma 4 and prove the first statement.

For the second statement, we adapt the result for the variant of the $(\mu + 1)$ GA which does not evaluate copies of either parents. When there is no diversity in the population the offspring is identical to the parent with probability p_0. Then, given that a fitness evaluation occurs, the probability of improvement via mutation is

$p_{0,m}/(1-p_0)$ and the probability that diversity is introduced is $p_{0,1}/(1-p_0)$. The proof is identical to the proof of the first statement, except for using probabilities $p^*_{0,m} = p_{0,m}/(1-p_0)$ and $p^*_{0,1} = p_{0,1}/(1-p_0)$ instead of $p_{0,1}$ and $p_{0,m}$ from Definition 1. Even if we pessimistically assume that a function evaluation occurs at every iteration when there is diversity in the population, we still get a $(1-p_0)$ decrease in the leading constant. □

Whether the copies of the parents are evaluated or not does not affect the algorithm's trajectory in the search space. Its effect on the expected runtime in turn can be estimated in a straightforward manner if the probability of producing a copy is known. In general this probability is a function of the mutation operator and the Hamming distances between each pair of individuals in the population. For the special case of monotypic populations (where the Hamming distance between all pairs is zero) it is equal to the probability that the mutation operator does not flip any bits, i.e., p_0. The probability of creating a copy decreases with the diversity of the population. However, for population sizes of $o(\sqrt{\log n})$, the expected number of iterations with a diverse population is asymptotically smaller than the number of iterations with a monotypic population. This property allows us to adapt the leading constant of the upper bound on the expected runtime of the standard algorithm by multiplying it with $(1-p_0)$.

5 Conclusion

In this work, we have shown that the steady-state $(\mu+1)$ GA optimises OneMax faster than any unary unbiased search heuristic. Providing precise asymptotic bounds on the expected runtime of standard GAs without artificial mechanisms that simplify the analysis has been a long standing open problem [22, 31]. We have derived bounds up to the leading term constants of the expected runtime. To achieve this result we show that a simplified Markov chain pessimistically represents the behaviour of the GA for OneMax. This insight about the algorithm/problem pair allows the derivation of runtime bounds for a complex multi-dimensional stochastic process. The analysis shows that as the number of states in the Markov chain (the population size) increases, so does the probability that diversity in the population is kept. Thus, larger populations increase the probability that recombination finds improved solutions quickly, hence reduce the expected runtime. Recent work has provided lower bounds on the expected runtime of the $(2+1)$ GA with mutation rates up to $1.422/n$ [27]. Future work should focus on deriving corresponding lower bounds for different population sizes to fill in the gaps left in this paper regarding the power of populations for hill-climbing OneMax-like functions. The only other lower bounds available for standard genetic algorithms are those derived to show that generational GAs are inefficient even for easy problems if fitness proportional selection is used [24, 25].

Acknowledgements This work was supported by EPSRC under Grant EP/M004252/1.

References

1. Corus, D., Dang, D.C., Eremeev, A.V., Lehre, P.K.: Level-based analysis of genetic algorithms and other search processes. IEEE Trans. Evolut. Comput. **22**(5), 707–719 (2018)
2. Corus, D., Oliveto, P.S.: Standard steady state genetic algorithms can hillclimb faster than mutation-only evolutionary algorithms. IEEE Trans. Evolut. Comput. **22**(5), 720–732 (2018)
3. Corus, D., Oliveto, P.S.: On the benefits of populations for the exploitation speed of standard steady-state genetic algorithms. In: Proceedings of the Genetic and Evolutionary Computation Conference, Proc. of GECCO'19, pp. 1452–1460 (2019)
4. Dang, D.C., Friedrich, T., Kötzing, T., Krejca, M.S., Lehre, P.K., Oliveto, P.S., Sudholt, D., Sutton, A.M.: Escaping local optima using crossover with emergent diversity. IEEE Trans. Evolut. Comput. **22**(3), 484–497 (2018)
5. Doerr, B., Happ, E., Klein, C.: Crossover can provably be useful in evolutionary computation. Theor. Comput. Sci. **425**, 17–33 (2012)
6. Doerr, B., Doerr, C.: Optimal parameter choices through self-adjustment: applying the 1/5-th rule in discrete settings. In: Proceedings of GECCO'15, pp. 1335–1342 (2015)
7. Doerr, B., Doerr, C.: A tight runtime analysis of the (1+(λ, λ)) genetic algorithm on OneMax. In: Proceedings of GECCO'15, pp. 1423–1430 (2015)
8. Doerr, B., Doerr, C., Yang, J.: Optimal parameter choices via precise black-box analysis. In: Proceedings of GECCO'16, pp. 1123–1130 (2016)
9. Eiben, A.E., Smith, J.E.: Introduction to Evolutionary Computing. Springer, Berlin (2003)
10. Friedrich, T., Oliveto, P.S., Sudholt, D., Witt, C.: Analysis of diversity-preserving mechanisms for global exploration. Evolut. Comput. **17**(4), 455–476 (2009)
11. He, J., Yao, X.: Towards an analytic framework for analysing the computation time of evolutionary algorithms. Artif. Intell. **145**(1–2), 59–97 (2003)
12. He, J., Yao, X.: A study of drift analysis for estimating computation time of evolutionary algorithms. Nat. Comput. **3**, 21–35 (2004)
13. Jansen, T.: Analyzing Evolutionary Algorithms: The Computer Science Perspective. Springer, Berlin (2013)
14. Jansen, T., Wegener, I.: The analysis of evolutionary algorithms-a proof that crossover really can help. Algorithmica **34**(1), 47–66 (2002)
15. Lehre, P.K., Oliveto, P.S.: Theoretical analysis of stochastic search algorithms. In: Resende, R.M.M.G.C., Pardalos, P.M. (eds.) Handbook of Heuristics. Springer, Berlin (2018)
16. Lehre, P.K., Witt, C.: Black-box search by unbiased variation. Algorithmica **64**(4), 623–642 (2012)
17. Lehre, P.K., Yao, X.: Crossover can be constructive when computing unique input-output sequences. Soft Comput. **15**(9), 1675–1687 (2011)
18. Lengler, J.: Drift analysis. In: Doerr, B., Neumann, F. (eds.) Theory of Evolutionary Computation: Recent Developments in Discrete Optimization, pp. 89–131. Springer, Berlin (2020)
19. Lengler, J.: A general dichotomy of evolutionary algorithms on monotone functions. In: Proceedings of PPSN XV, pp. 3–15. Springer (2018)
20. Lengler, J., Zou, X.: Exponential slowdown for larger populations: The (μ + 1)-ea on monotone functions. In: Proceedings of FOGA, pp. 87–101 (2019)
21. Li, H.B., Huang, T.Z., Liu, X.P., Li, H.: On the inverses of general tridiagonal matrices. Linear Algebra Appl. **433**(5), 965–983 (2010)

22. Mitchell, M., Holland, J.H., Forrest, S.: When will a genetic algorithm outperform hill climbing. In: Cowan, J.D., Tesauro, G., Alspector, J. (eds.) Advances in Neural Information Processing Systems, vol. 6, pp. 51–58. Morgan Kaufmann Publishers, Burlington (1994)
23. Neumann, F., Oliveto, P.S., Rudolph, G., Sudholt, D.: On the effectiveness of crossover for migration in parallel evolutionary algorithms. In: Proceedings of GECCO'11, pp. 1587–1594 (2011)
24. Oliveto, P.S., Witt, C.: On the runtime analysis of the simple genetic algorithm. Theor. Comput. Sci. **545**, 2–19 (2014)
25. Oliveto, P.S., Witt, C.: Improved time complexity analysis of the simple genetic algorithm. Theor. Comput. Sci. **605**, 21–41 (2015)
26. Oliveto, P.S., Yao, X.: Runtime analysis of evolutionary algorithms for discrete optimization. In: Doerr, B., Auger, A. (eds.) Theory of Randomized Search Heuristics: Foundations and Recent Developments, p. 21. World Scientific, Singapore (2011)
27. Oliveto, P.S., Sudholt, D., Witt, C.: A tight lower bound on the expected runtime ofstandard steady state genetic algorithms. In: Proceedings of GECCO'20, p. 1323–1331 (2020)
28. Pinto, E.C., Doerr, C.: A simple proof for the usefulness of crossover in black-box optimization. In: Proceedings of PPSN XV, pp. 29–41 (2018)
29. Sarma, J., Jong, K.D.: Generation gap methods. In: Back, T., Fogel, D.B., Michalewicz, Z. (eds.) Handbook of Evolutionary Computation. IOP Publishing Ltd, Bristol (1997)
30. Sudholt, D.: A new method for lower bounds on the running time of evolutionary algorithms. IEEE Trans. Evolut. Comput. **17**(3), 418–435 (2013)
31. Sudholt, D.: How crossover speeds up building block assembly in genetic algorithms. Evolut. Comput. **25**(2), 237–274 (2017)
32. Sudholt, D.: Crossover is provably essential for the ising model on trees. In: Proceedings of GECCO'11, pp. 1161–1167. New York, New York, USA (2005)
33. Sutton, A.: Crossover can simulate bounded tree search on a fixed-parameter tractable optimization problem. In: Proceedings of GECCO'18, pp. 1531–1538 (2018)
34. Witt, C.: Runtime analysis of the $(\mu + 1)$ ea on simple pseudo-boolean functions. Evolut. Comput. **14**(1), 65–86 (2006)
35. Witt, C.: Tight bounds on the optimization time of a randomized search heuristic on linear functions. Combin. Probab. Comput. **22**(2), 294–318 (2013)
36. Yu, Y., Qian, C., Zhou, Z.H.: Switch analysis for running time analysis of evolutionary algorithms. IEEE Trans. Evolut. Comput. **19**(6), 777–792 (2015)